**Sprawozdanie**

*Systemy wbudowane*



**Ćwiczenie 6:**  
Układy czasowo-licznikowe.

Wykonanie:

**Busłowski Tomasz**

**Suchwałko Tomasz**

**Skrouba Kamil**

**Zawadzka Magdalena  
(Grupa PS3)**

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

Zadania do wykonania

1. Napisz program sterujący 4 diodami LED z różną częstotliwością (0,25Hz,0,5Hz,1Hzi 2Hz) wykorzystując do tego TIM1 i kanały CC
2. Napisz stoper mierzący 4 różne czasy z dokładnością do 0,01 sek. Czasy mają być wyświetlane jednocześnie na wyświetlaczu LCD w formacie 00:00 (sek : 1/100 sek). Poszczególne stopery mają być uruchamiane i zatrzymywane oddzielnymi przyciskami (START/STOP).
3. Napisz program sterujący poziomem jasności diody LED przy pomocy joysticka (w górę – jaśniej, w dół – ciemniej).
4. Napisz program sterujący sygnalizatorem dźwiękowym (złącze Con7, pin SPK) na płytce umożliwiający odtwarzanie melodii. Sygnał sterujący głośnikiem można generować z wykorzystaniem sygnału PWM o wypełnieniu 50%. Zaprezentować działanie programu na przykładzie dowolnej melodii (np. „Wlazł kotek na płotek”, „Hej sokoły”, itp.).

Zadanie 1

**Treść:**

Napisz program sterujący 4 diodami LED z różną częstotliwością (0,25Hz,0,5Hz,1Hz i 2Hz) wykorzystując do tego TIM1 i kanały CC

**Realizacja:**

#include "stm32f10x.h"

#include "stm32f10x\_tim.h"

void GPIO\_Config(void);

void RCC\_Config(void);

void NVIC\_Config(void);

void TIM\_Config(void);

void TIM1\_UP\_IRQHandler(void);

unsigned int counter = 0;

int main(void) {

RCC\_Config();

GPIO\_Config();

NVIC\_Config();

TIM\_Config();

GPIO\_ResetBits(GPIOB, GPIO\_Pin\_0 | GPIO\_Pin\_1 | GPIO\_Pin\_2 | GPIO\_Pin\_3);

while (1) {};

return 0;

}

void RCC\_Config(void)

{

ErrorStatus HSEStartUpStatus;

RCC\_DeInit(); //Resetetowanie ustawien RCC

RCC\_HSEConfig(RCC\_HSE\_ON); // Wlaczenie HSE

HSEStartUpStatus = RCC\_WaitForHSEStartUp();//OdczekajazHSE bedziegotowy

if (HSEStartUpStatus == SUCCESS)

{

FLASH\_PrefetchBufferCmd(FLASH\_PrefetchBuffer\_Enable);

FLASH\_SetLatency(FLASH\_Latency\_2);//ustaw opoznienie Pamieci Flash zalezni eod taktowania rdzenia //0 <24MHz;1:24~48MHz;2:>48MHz

RCC\_HCLKConfig(RCC\_SYSCLK\_Div1); //ustaw HCLK=SYSCLK

RCC\_PCLK2Config(RCC\_HCLK\_Div1); //ustaw PCLK2=HCLK

RCC\_PCLK1Config(RCC\_HCLK\_Div2); //ustaw PCLK1=HCLK/2

RCC\_PLLConfig(RCC\_PLLSource\_HSE\_Div1, RCC\_PLLMul\_9); //ustaw PLLCLK= HSE\*9 czyli8MHz\* 9= 72 MHz

RCC\_PLLCmd(ENABLE); //wlaczPLL

while (RCC\_GetFlagStatus(RCC\_FLAG\_PLLRDY) == RESET); //odczekaj na poprawne uruchomienie PLL

RCC\_SYSCLKConfig(RCC\_SYSCLKSource\_PLLCLK); //ustaw PLL jako zrodlo sygnalu zegarowego

while (RCC\_GetSYSCLKSource() != 0x08); //odczekaj az PLL bedzie sygnalem zegarowym systemu

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_GPIOB,ENABLE); //wlacztaktowanieportu GPIOB

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_TIM1,ENABLE);

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_GPIOA | RCC\_APB2Periph\_GPIOB |

RCC\_APB2Periph\_AFIO, ENABLE); //wlacztaktowanieportu GPIO A

}

else {}

}

void NVIC\_Config(void) {

NVIC\_InitTypeDef NVIC\_InitStructure; //Konfigurowanie kontrolera przerwan NVIC

#ifdef VECT\_TAB\_RAM // Jezeli tablica wektorow w RAM, to ustaw jejadresna0x20000000

NVIC\_SetVectorTable(NVIC\_VectTab\_RAM, 0x0);

#else // VECT\_TAB\_FLASH // W przeciwnym wypadku ustaw na0x08000000

NVIC\_SetVectorTable(NVIC\_VectTab\_FLASH, 0x0); #endif

NVIC\_PriorityGroupConfig(NVIC\_PriorityGroup\_1); //przerwanie UP (przepelnienie)timera1

NVIC\_InitStructure.NVIC\_IRQChannel = TIM1\_UP\_IRQn;

NVIC\_InitStructure.NVIC\_IRQChannelPreemptionPriority = 0;

NVIC\_InitStructure.NVIC\_IRQChannelSubPriority = 1;

NVIC\_InitStructure.NVIC\_IRQChannelCmd = ENABLE;

NVIC\_Init( & NVIC\_InitStructure);

}

void GPIO\_Config(void) { //konfigurowanie portow GPIO

GPIO\_InitTypeDef GPIO\_InitStructure; // disable JTAG

GPIO\_PinRemapConfig(GPIO\_Remap\_SWJ\_JTAGDisable, ENABLE); /\*Tu nalezy umiescic kod zwiazany zkonfiguracjaposzczegolnych portow

GPIOpotrzebnych wprogramie\*/

GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_0 | GPIO\_Pin\_1 |

GPIO\_Pin\_2 | GPIO\_Pin\_3;

GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_Out\_PP;

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

}

void TIM\_Config(void) { //Konfiguracja timerow

TIM\_TimeBaseInitTypeDef TIM\_TimeBaseStructure;

TIM\_OCInitTypeDef TIM\_OCInitStructure;

//Konfiguracja licznika 1

//Ustawienia taktowaniaitrybu pracy licznika1

TIM\_TimeBaseStructure.TIM\_Prescaler = 7200 - 1;

//taktowanielicznkafclk = 72MHz/7200 =10 kHz

TIM\_TimeBaseStructure.TIM\_Period = 2500;

//okresprzepelnien licznika= 20000 taktow= 2sekundy

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

TIM\_TimeBaseStructure.TIM\_RepetitionCounter = 0;

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

TIM\_TimeBaseInit(TIM1, & TIM\_TimeBaseStructure);

TIM\_ITConfig(TIM1, TIM\_IT\_Update, ENABLE); // Wlaczenie timerow

Obsługa Przerwań:

void TIM1\_UP\_IRQHandler(void) {

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

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

if (counter % 4 == 0)

GPIO\_WriteBit(GPIOB, GPIO\_Pin\_3, (BitAction)(1 -

GPIO\_ReadOutputDataBit(GPIOB, GPIO\_Pin\_3)));

if (counter % 3 == 0)

GPIO\_WriteBit(GPIOB, GPIO\_Pin\_2, (BitAction)(1 -

GPIO\_ReadOutputDataBit(GPIOB, GPIO\_Pin\_2)));

if (counter % 2 == 0)

GPIO\_WriteBit(GPIOB, GPIO\_Pin\_1, (BitAction)(1 -

GPIO\_ReadOutputDataBit(GPIOB, GPIO\_Pin\_1)));

if (counter % 1 == 0)

GPIO\_WriteBit(GPIOB, GPIO\_Pin\_0, (BitAction)(1 -

GPIO\_ReadOutputDataBit(GPIOB, GPIO\_Pin\_0)));

}

counter++;

}

}

Zadanie 2

**Treść:**

Napisz stoper mierzący 4 różne czasy z dokładnością do 0,01 sek. Czasy mają być wyświetlane jednocześnie na wyświetlaczu LCD w formacie 00:00 (sek : 1/100 sek). Poszczególne stopery mają być uruchamiane i zatrzymywane oddzielnymi przyciskami (START/STOP).

**Realizacja:**

#include "stm32f10x.h"

#include "stm32f10x\_tim.h"

#include "stm32f10x\_exti.h"

#include "lcd.h"

#include <stdio.h>

void GPIO\_Config(void);

void RCC\_Config(void);

void NVIC\_Config(void);

void TIM\_Config(void);

int STOPER1\_ON=0,STOPER2\_ON=0,STOPER3\_ON=0,STOPER4\_ON=0;

int time\_s1=0, time\_s2=0, time\_s3=0, time\_s4=0;

char t1[] = " "; char t2[] = " "; char t3[] = " "; char t4[] = " ";

char tt1[] = " "; char tt2[] = " "; char tt3[] = " "; char tt4[] = " ";

int s1, ss1;

int s2, ss2;

int s3, ss3;

int s4, ss4;

int temp;

int main(void)

{

//konfiguracja systemu

RCC\_Config();

GPIO\_Config();

NVIC\_Config();

LCD\_Initialize();

TIM\_Config();

LCD\_WriteCommand(LCD\_CLEAR); while (1) { }; return 0;

}

void RCC\_Config(void)

//konfigurowanie sygnalow taktujacych

{

ErrorStatus

HSEStartUpStatus;//zmiennaopisujacarezultaturuchomieniaHSE

RCC\_DeInit(); //Resetustawien RCC

RCC\_HSEConfig(RCC\_HSE\_ON);//WlaczenieHSE

HSEStartUpStatus = RCC\_WaitForHSEStartUp(); //OdczekajazHSE bedziegotowy

if (HSEStartUpStatus == SUCCESS) {

FLASH\_PrefetchBufferCmd(FLASH\_PrefetchBuffer\_Enable);

FLASH\_SetLatency(FLASH\_Latency\_2);//ustaw opoznienie Pamieci Flash zalezni eod taktowania rdzenia //0 <24MHz;1:24~48MHz;2:>48MHz

RCC\_HCLKConfig(RCC\_SYSCLK\_Div1); //ustaw HCLK=SYSCLK

RCC\_PCLK2Config(RCC\_HCLK\_Div1); //ustaw PCLK2=HCLK

RCC\_PCLK1Config(RCC\_HCLK\_Div2); //ustaw PCLK1=HCLK/2

RCC\_PLLConfig(RCC\_PLLSource\_HSE\_Div1, RCC\_PLLMul\_9); //ustaw PLLCLK= HSE\*9 czyli8MHz\* 9= 72 MHz

RCC\_PLLCmd(ENABLE); //wlaczPLL

while (RCC\_GetFlagStatus(RCC\_FLAG\_PLLRDY) == RESET); //odczekaj na poprawne uruchomienie PLL

RCC\_SYSCLKConfig(RCC\_SYSCLKSource\_PLLCLK); //ustaw PLL jako zrodlo sygnalu zegarowego

while (RCC\_GetSYSCLKSource() != 0x08); //odczekaj az PLL bedzie sygnalem zegarowym systemu

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_GPIOB,ENABLE); //wlacztaktowanieportu GPIOB

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_TIM1,ENABLE);

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_GPIOA | RCC\_APB2Periph\_GPIOB |

RCC\_APB2Periph\_AFIO, ENABLE); //wlacztaktowanieportu GPIO A

}

else {}

}

void NVIC\_Config(void) {

NVIC\_InitTypeDef NVIC\_InitStructure;

NVIC\_InitTypeDef NVIC\_InitStructure1;

NVIC\_InitTypeDef NVIC\_InitStructure2;

NVIC\_InitTypeDef NVIC\_InitStructure3;

NVIC\_InitTypeDef NVIC\_InitStructure4;

EXTI\_InitTypeDef EXTI\_InitStructure;

EXTI\_InitTypeDef EXTI\_InitStructure1;

EXTI\_InitTypeDef EXTI\_InitStructure2;

EXTI\_InitTypeDef EXTI\_InitStructure3; //Konfigurowanie kontrolera przerwan NVIC

#ifdef VECT\_TAB\_RAM // Jezeli tablica wektorow w RAM, to ustaw adres na 0x20000000

NVIC\_SetVectorTable(NVIC\_VectTab\_RAM, 0x0);

#else // VECT\_TAB\_FLASH

// W przeciwnym wypadku ustaw na0x08000000

NVIC\_SetVectorTable(NVIC\_VectTab\_FLASH, 0x0);

#endif

NVIC\_PriorityGroupConfig(NVIC\_PriorityGroup\_1);

//przerwanie UP (przepelnienie)timera1

NVIC\_InitStructure.NVIC\_IRQChannel= TIM1\_UP\_IRQn;

NVIC\_InitStructure.NVIC\_IRQChannelPreemptionPriority = 1;

NVIC\_InitStructure.NVIC\_IRQChannelSubPriority = 1;

NVIC\_InitStructure.NVIC\_IRQChannelCmd = ENABLE;

NVIC\_Init(&NVIC\_InitStructure);

NVIC\_InitStructure1.NVIC\_IRQChannel= EXTI0\_IRQn;

//Wyborkonfigurowanego IRQ

NVIC\_InitStructure1.NVIC\_IRQChannelPreemptionPriority = 0;

//Priorytetgrupowy

NVIC\_InitStructure1.NVIC\_IRQChannelSubPriority = 1;

//Podpriorytet

NVIC\_InitStructure1.NVIC\_IRQChannelCmd = ENABLE;

//WlaczenieobslugiIRQ

NVIC\_Init(&NVIC\_InitStructure1);

NVIC\_InitStructure2.NVIC\_IRQChannel= EXTI1\_IRQn;

//Wyborkonfigurowanego IRQ

NVIC\_InitStructure2.NVIC\_IRQChannelPreemptionPriority = 0;

//Priorytetgrupowy

NVIC\_InitStructure2.NVIC\_IRQChannelSubPriority = 1;

//Podpriorytet

NVIC\_InitStructure2.NVIC\_IRQChannelCmd = ENABLE;

//WlaczenieobslugiIRQ

NVIC\_Init(&NVIC\_InitStructure2);

NVIC\_InitStructure3.NVIC\_IRQChannel= EXTI2\_IRQn;

//Wyborkonfigurowanego IRQ

NVIC\_InitStructure3.NVIC\_IRQChannelPreemptionPriority = 0;

//Priorytetgrupowy

NVIC\_InitStructure3.NVIC\_IRQChannelSubPriority = 1;

//Podpriorytet

NVIC\_InitStructure3.NVIC\_IRQChannelCmd = ENABLE;

//WlaczenieobslugiIRQ

NVIC\_Init(&NVIC\_InitStructure3);

NVIC\_InitStructure4.NVIC\_IRQChannel= EXTI3\_IRQn;

//Wyborkonfigurowanego IRQ

NVIC\_InitStructure4.NVIC\_IRQChannelPreemptionPriority = 0;

//Priorytetgrupowy

NVIC\_InitStructure4.NVIC\_IRQChannelSubPriority = 1; //Podpriorytet

NVIC\_InitStructure4.NVIC\_IRQChannelCmd = ENABLE;

//WlaczenieobslugiIRQ

NVIC\_Init(&NVIC\_InitStructure4);

GPIO\_EXTILineConfig(GPIO\_PortSourceGPIOA, GPIO\_PinSource0); //Ustawieniezrodlaprzerwania

GPIO\_EXTILineConfig(GPIO\_PortSourceGPIOA, GPIO\_PinSource1); GPIO\_EXTILineConfig(GPIO\_PortSourceGPIOA, GPIO\_PinSource2); //Ustawieniezrodlaprzerwania

GPIO\_EXTILineConfig(GPIO\_PortSourceGPIOA, GPIO\_PinSource3);

//Konfiguracja przerwania EXTI0 nalinie0

EXTI\_InitStructure.EXTI\_Line= EXTI\_Line0;//Wyborlinii

EXTI\_InitStructure.EXTI\_Mode= EXTI\_Mode\_Interrupt;

//Ustawieniegenerowaniaprzerwania(aniezdarzenia)

EXTI\_InitStructure.EXTI\_Trigger= EXTI\_Trigger\_Falling;

//Wyzwalaniezboczemopadajacym(wcisniecieprzycisku)

EXTI\_InitStructure.EXTI\_LineCmd = ENABLE;

//Wlaczenieprzerwania

EXTI\_Init(&EXTI\_InitStructure);

EXTI\_InitStructure1.EXTI\_Line= EXTI\_Line1;//Wyborlinii

EXTI\_InitStructure1.EXTI\_Mode= EXTI\_Mode\_Interrupt;

//Ustawieniegenerowaniaprzerwania(aniezdarzenia)

EXTI\_InitStructure1.EXTI\_Trigger= EXTI\_Trigger\_Falling;

//Wyzwalaniezboczemopadajacym(wcisniecieprzycisku)

EXTI\_InitStructure1.EXTI\_LineCmd

ENABLE;//Wlaczenieprzerwania

EXTI\_Init(&EXTI\_InitStructure1);

EXTI\_InitStructure2.EXTI\_Line= EXTI\_Line2;//Wyborlinii

EXTI\_InitStructure2.EXTI\_Mode= EXTI\_Mode\_Interrupt;

//Ustawieniegenerowaniaprzerwania(aniezdarzenia)

EXTI\_InitStructure2.EXTI\_Trigger= EXTI\_Trigger\_Falling;

//Wyzwalaniezboczemopadajacym(wcisniecieprzycisku) =

EXTI\_InitStructure2.EXTI\_LineCmd

ENABLE;//Wlaczenieprzerwania

EXTI\_Init(&EXTI\_InitStructure2);

EXTI\_InitStructure3.EXTI\_Line= EXTI\_Line3;//Wybor linii

EXTI\_InitStructure3.EXTI\_Mode= EXTI\_Mode\_Interrupt; =

//Ustawieniegenerowaniaprzerwania(aniezdarzenia)

EXTI\_InitStructure3.EXTI\_Trigger= EXTI\_Trigger\_Falling;

//Wyzwalaniezboczemopadajacym(wcisniecieprzycisku)

EXTI\_InitStructure3.EXTI\_LineCmd = ENABLE;

//Wlaczenieprzerwania

EXTI\_Init(&EXTI\_InitStructure3);

}

void GPIO\_Config(void) { //konfigurowanie portow GPIO

GPIO\_InitTypeDef GPIO\_InitStructure;

// disable JTAG

GPIO\_PinRemapConfig(GPIO\_Remap\_SWJ\_JTAGDisable, ENABLE);

GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_0 |GPIO\_Pin\_1

|GPIO\_Pin\_2 |GPIO\_Pin\_3;

GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;

GPIO\_InitStructure.GPIO\_Mode= GPIO\_Mode\_IN\_FLOATING;

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

GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_0 |GPIO\_Pin\_1

|GPIO\_Pin\_3 |GPIO\_Pin\_4|

GPIO\_Pin\_5 | GPIO\_Pin\_7 |

GPIO\_Pin\_8;

GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;

GPIO\_InitStructure.GPIO\_Mode= GPIO\_Mode\_IN\_FLOATING;

//GPIO\_Init(GPIOB, &GPIO\_InitStructure);

}

void TIM\_Config(void) {

//Konfiguracja timerow

TIM\_TimeBaseInitTypeDef TIM\_TimeBaseStructure;

TIM\_OCInitTypeDef TIM\_OCInitStructure;

//Konfiguracja licznika 1

//Ustawienia taktowaniaitrybu pracy licznika1

TIM\_TimeBaseStructure.TIM\_Prescaler=

7200-1;//taktowanielicznkafclk = 72MHz/7200 =10kHz

TIM\_TimeBaseStructure.TIM\_Period = 100;//okresprzepelnien licznika= 100 taktow= 1setna sekundy

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

TIM\_TimeBaseStructure.TIM\_RepetitionCounter=0;

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

TIM\_TimeBaseInit(TIM1, &TIM\_TimeBaseStructure);

TIM\_ITConfig(TIM1, TIM\_IT\_Update, ENABLE);

// Wlaczenie timerow

TIM\_Cmd(TIM1, ENABLE);

}

void EXTI0\_IRQHandler(void)

{

if (EXTI\_GetITStatus(EXTI\_Line0)!= RESET) {

int delay = 0;

for(delay=0; delay<1000; delay++);

STOPER1\_ON = (STOPER1\_ON+1)%2;

EXTI\_ClearITPendingBit(EXTI\_Line0);

}

}

void EXTI1\_IRQHandler(void)

{

if (EXTI\_GetITStatus(EXTI\_Line1)!= RESET) {

int delay = 0;

for(delay=0; delay<1000; delay++);

STOPER1\_ON = (STOPER2\_ON+1)%2;

EXTI\_ClearITPendingBit(EXTI\_Line1);

}

}

void EXTI1\_IRQHandler(void)

{

if (EXTI\_GetITStatus(EXTI\_Line2)!= RESET) {

int delay = 0;

for(delay=0; delay<1000; delay++);

STOPER1\_ON = (STOPER3\_ON+1)%2;

EXTI\_ClearITPendingBit(EXTI\_Line2);

}

}

void EXTI1\_IRQHandler(void)

{

if (EXTI\_GetITStatus(EXTI\_Line3)!= RESET) {

int delay = 0;

for(delay=0; delay<1000; delay++);

STOPER1\_ON = (STOPER4\_ON+1)%2;

EXTI\_ClearITPendingBit(EXTI\_Line3);

}

}

void TIM1\_UP\_IRQHandler(void)

{

int i;

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

{

if(STOPER1\_ON ==0)time\_s1 +=1;

if(STOPER2\_ON ==0)time\_s2 +=1;

if(STOPER3\_ON ==0)time\_s3 +=1;

if(STOPER4\_ON ==0)time\_s4 +=1;

for(i=0;i<8;i++){

t1[i]=' '; t2[i]=' '; t3[i]=' '; t4[i]=' '; }

s1= time\_s1/100; ss1 = time\_s1%100;

s2= time\_s2/100; ss2 = time\_s2%100;

s3= time\_s3/100; ss3 = time\_s3%100;

s4= time\_s4/100; ss4 = time\_s4%100;

sprintf(t1,"T1 %i:%i",s1,ss1);

sprintf(t2,"T2 %i:%i",s2,ss2);

sprintf(t3,"T3 %i:%i",s3,ss3);

sprintf(t4,"T4 %i:%i",s4,ss4);

LCD\_WriteCommand(LCD\_HOME);

LCD\_WriteString(t1,8);

LCD\_WriteString(t2,8);

for(i = 0; i < 5000; i++){}

LCD\_WriteCommand(0xC0);

or(i = 0; i < 5000; i++){}

LCD\_WriteString(t3,8);

LCD\_WriteString(t4,8);

}

}

Zadanie 3

**Treść:**

Napisz program sterujący poziomem jasności diody LED przy pomocy joysticka (w górę – jaśniej, w dół – ciemniej).

**Realizacja:**

#include "stm32f10x.h"

#include "stm32f10x\_tim.c"

void GPIO\_Config(void);

void RCC\_Config(void);

void NVIC\_Config(void);

void TIM\_Config(void);

TIM\_OCInitTypeDef TIM\_OCInitStructure;

int main(void)

{ volatile unsigned long int i;

uint8\_t button\_state=0xFF, temp=0, port\_data ;

GPIO\_ResetBits(GPIOA, GPIO\_Pin\_0 | GPIO\_Pin\_1 | GPIO\_Pin\_2 |

GPIO\_Pin\_3);

GPIO\_ResetBits(GPIOB, GPIO\_Pin\_0 | GPIO\_Pin\_1 | GPIO\_Pin\_2 |

GPIO\_Pin\_3 | GPIO\_Pin\_4 | GPIO\_Pin\_5 | GPIO\_Pin\_6 | GPIO\_Pin\_7); //konfiguracja systemu

RCC\_Config();

GPIO\_Config();

NVIC\_Config();

TIM\_Config();

for(i = 0; i < 0x2500ul; i++);

for(i = 0; i < 0x2500ul; i++);

for(i = 0; i < 0x2500ul; i++);

while (1) {

port\_data = GPIO\_ReadInputData(GPIOA); //czytaj port GPIOA

for(i = 0; i < 0x2500ul; i++);

temp = port\_data ^ button\_state;

temp &= button\_state;

button\_state = port\_data;

if (temp & 0x01){

TIM\_OCInitStructure.TIM\_Pulse = TIM\_OCInitStructure.TIM\_Pulse + 2500ul;

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

}

if (temp & 0x02){

TIM\_OCInitStructure.TIM\_Pulse = TIM\_OCInitStructure.TIM\_Pulse - 2500ul;

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

} };

return 0;

}

void RCC\_Config(void)

//konfigurowanie sygnalow taktujacych

{

ErrorStatus HSEStartUpStatus;

//zmienna opisujaca rezultat uruchomienia HSE

RCC\_DeInit(); //Reset ustawien RCC

RCC\_HSEConfig(RCC\_HSE\_ON);

//Wlaczenie HSE

HSEStartUpStatus = RCC\_WaitForHSEStartUp();

//Odczekaj az HSE bedzie gotowy if(HSEStartUpStatus == SUCCESS) {

FLASH\_PrefetchBufferCmd(FLASH\_PrefetchBuffer\_Enable);//

FLASH\_SetLatency(FLASH\_Latency\_2); //ustaw

zwloke dla pamieci Flash; zaleznie od taktowania rdzenia

//0:<24MHz; 1:24~48MHz; 2:>48MHz

RCC\_HCLKConfig(RCC\_SYSCLK\_Div1); //ustaw

HCLK=SYSCLK

RCC\_PCLK2Config(RCC\_HCLK\_Div1); //ustaw

PCLK2=HCLK

RCC\_PCLK1Config(RCC\_HCLK\_Div2); //ustaw

PCLK1=HCLK/2

RCC\_PLLConfig(RCC\_PLLSource\_HSE\_Div1, RCC\_PLLMul\_9); //ustaw

PLLCLK = HSE\*9 czyli 8MHz \* 9 = 72 MHz

RCC\_PLLCmd(ENABLE); //wlacz PLL

while(RCC\_GetFlagStatus(RCC\_FLAG\_PLLRDY)

//odczekaj na poprawne uruchomienie PLL == RESET);

RCC\_SYSCLKConfig(RCC\_SYSCLKSource\_PLLCLK); PLL jako zrodlo sygnalu zegarowego //ustaw

while(RCC\_GetSYSCLKSource() != 0x08);

//odczekaj az PLL bedzie sygnalem zegarowym systemu /\*Tu nalezy umiescic kod zwiazany z konfiguracja sygnalow zegarowych potrzebnych w programie peryferiow\*/

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_GPIOB, ENABLE);//wlacz taktowanie portu GPIO B

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_GPIOA,

ENABLE);//wlacz taktowanie portu GPIO A

RCC\_APB1PeriphClockCmd(RCC\_APB1Periph\_TIM4, ENABLE); //wlacz taktowanie licznika TIM4

} else {

}

}

void NVIC\_Config(void) {

//Konfigurowanie kontrolera przerwan NVIC

#ifdef VECT\_TAB\_RAM

// Jezeli tablica wektorow w RAM, to ustaw jej adres na 0x20000000

NVIC\_SetVectorTable(NVIC\_VectTab\_RAM, 0x0);

#else // VECT\_TAB\_FLASH

// W przeciwnym wypadku ustaw na 0x08000000

NVIC\_SetVectorTable(NVIC\_VectTab\_FLASH, 0x0); #endif

}

void GPIO\_Config(void) { //konfigurowanie portow GPIO

GPIO\_InitTypeDef GPIO\_InitStructure;

/\*Tu nalezy umiescic kod zwiazany z konfiguracja poszczegolnych portow GPIO potrzebnych w programie\*/

GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_8 | GPIO\_Pin\_9 ;

GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;

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

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

GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_0 | GPIO\_Pin\_1 | GPIO\_Pin\_2 | GPIO\_Pin\_3;

GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_IN\_FLOATING;

//wejscie bez podciagania

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

}

void TIM\_Config(void) { //konfigurowanie licznikow

TIM\_TimeBaseInitTypeDef TIM\_TimeBaseStructure;

//TIM\_OCInitTypeDef TIM\_OCInitStructure;

// Konfiguracja TIM4

// Ustawienia ukladu podstawy czasu

TIM\_TimeBaseStructure.TIM\_Prescaler = 0; prescelera (prescaler=1) //bez

TIM\_TimeBaseStructure.TIM\_Period = 49999ul; //czestotliwosc

PWM = 1440 Hz

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 = 5000ul; //wypelnienie = 50000/5000=10%

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

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

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

// Konfiguracja kanalu 4 - uzywamy kanalu 4 poniewaz jego

wyjscie jest na GPIOB9 - gdzie jest LED2

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

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

TIM\_OCInitStructure.TIM\_Pulse = 40000ul;; //wypelnienie = 50000/40000=80%

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

TIM\_OC4Init(TIM4, &TIM\_OCInitStructure);

TIM\_OC4PreloadConfig(TIM4, TIM\_OCPreload\_Enable);//wlaczenie buforowania

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

TIM\_Cmd(TIM4, ENABLE);

} Zadanie 4

**Treść:**

Napisz program sterujący sygnalizatorem dźwiękowym (złącze Con7, pin SPK) na płytce umożliwiający odtwarzanie melodii. Sygnał sterujący głośnikiem można generować z wykorzystaniem sygnału PWM o wypełnieniu 50%. Zaprezentować działanie programu na przykładzie dowolnej melodii (np. „Wlazł kotek na płotek”, „Hej sokoły”, itp.).

**Realizacja:**

#include "stm32f10x.h"

#include "stm32f10x\_exti.h"

#include "stm32f10x\_tim.h"

#define SysTick\_Frequency 9000000 //9MHz void GPIO\_Config(void); void RCC\_Config(void); void NVIC\_Config(void); void TIM\_Config(void); volatile int i = 0;

TIM\_OCInitTypeDef TIM\_OCInitStructure;

void sound(int Hz){

int onTime = 500000; int offTime = 5000;

TIM\_SetAutoreload(TIM4,60000/Hz);

TIM\_SetCompare1(TIM4,(60000/Hz)/2);

TIM\_Cmd(TIM4,ENABLE);

for(i = 0; i <onTime;i++){} TIM\_Cmd(TIM4,DISABLE); for(i = 0; i <offTime;i++){}

}

int main(void)

{

int delay = 500000; RCC\_Config();

GPIO\_Config();

NVIC\_Config(); TIM\_Config();

while (1) { //FFFFDFGAGFGF FFFFDAGFGFG

sound(350);

sound(350);

sound(350);

sound(350);

sound(294);

sound(350);.

sound(392);

sound(440);

sound(392);

sound(350);

sound(392);

sound(350);

for(i = 0; i<delay;i++){}

};

return 0;

}

void RCC\_Config(void)

{

ErrorStatus HSEStartUpStatus;

RCC\_DeInit();

RCC\_HSEConfig(RCC\_HSE\_ON);

HSEStartUpStatus = RCC\_WaitForHSEStartUp();

if(HSEStartUpStatus == SUCCESS){

FLASH\_PrefetchBufferCmd(FLASH\_PrefetchBuffer\_Enable);

FLASH\_SetLatency(FLASH\_Latency\_2);

RCC\_HCLKConfig(RCC\_SYSCLK\_Div1);

RCC\_PCLK2Config(RCC\_HCLK\_Div1);

RCC\_PCLK1Config(RCC\_HCLK\_Div2);

RCC\_PLLConfig(RCC\_PLLSource\_HSE\_Div1, RCC\_PLLMul\_9);

RCC\_PLLCmd(ENABLE);

while(RCC\_GetFlagStatus(RCC\_FLAG\_PLLRDY)== RESET);

RCC\_SYSCLKConfig(RCC\_SYSCLKSource\_PLLCLK);

while(RCC\_GetSYSCLKSource()!= 0x08);

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_GPIOA, ENABLE);

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_GPIOB, ENABLE);

RCC\_APB1PeriphClockCmd(RCC\_APB1Periph\_TIM4, ENABLE);

}

else { }

}

void NVIC\_Config(void) {

#ifdef VECT\_TAB\_RAM

NVIC\_SetVectorTable(NVIC\_VectTab\_RAM, 0x00);

#else // VECT\_TAB\_FLASH

NVIC\_SetVectorTable(NVIC\_VectTab\_FLASH, 0x00);

#endif

}

void GPIO\_Config(void) {

GPIO\_InitTypeDef GPIO\_InitStructure;

GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_8 ;

GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;

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

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

GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_0 |GPIO\_Pin\_1;

GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;

GPIO\_InitStructure.GPIO\_Mode= GPIO\_Mode\_IN\_FLOATING;

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

}

void TIM\_Config() {

TIM\_TimeBaseInitTypeDef TIM\_TimeBaseStructure;

TIM\_OCInitTypeDef TIM\_OCInitStructure;

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

TIM\_TimeBaseStructure.TIM\_Period = 170;

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

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

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

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

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

TIM\_OCInitStructure.TIM\_Pulse= 85;

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

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

TIM\_OC3PreloadConfig(TIM4, TIM\_OCPreload\_Enable);

TIM\_ARRPreloadConfig(TIM4, ENABLE);

TIM\_Cmd(TIM4, ENABLE);

}

**Kody źródłowe:**