**Sprawozdanie**

*Systemy wbudowane*



**Ćwiczenie 5:**  
Wyświetlacz alfanumeryczny LCD.

Wykonanie:

**Busłowski Tomasz**

**Suchwałko Tomasz**

**Skrouba Kamil**

**Zawadzka Magdalena  
(Grupa PS3)**

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

Zakres Materiału

1. Interfejs komunikacyjny wyświetlacza alfanumerycznego na sterowniku HD44780.
2. Tryby pracy interfejsu (4- i 8-bitowy).
3. Procedura inicjalizacji wyświetlacza.
4. Pamięć DDRAM i CGRAM.
5. Komendy sterujące pracą wyświetlacza.

Zadania do wykonania

1. Utwórz projekt, do którego należy dodać zawartość pliku cw5p1.zip. Uzupełnij program o brakujące parametry komend (plik lcd.h) oraz procedury inicjalizacji i wysyłania znaku do pamięci DDRAM (plik lcd.c). Zademonstruj działanie wyświetlacza wypisując napis np. „Hello World!”
2. Napisz program, który wypełni pamięć DDRAM znakami o kodach od 0x20 do 0x6F. Następnie każdorazowe wciśnięcie przycisku S1 ma przesunąć wyświetlacz w lewo, a przycisku S2 – w prawo.
3. Napisz na wyświetlaczu w pierwszym wierszu na środku słowo „Systemy”, a w drugim na środku słowo „Wbudowane”. Następnie spraw, aby cały wyświetlacz migał z okresem 1 sek.
4. Napisz program, który wyświetli na ekranie napis „Programowanie ARM jest łatwe”. Do wyświetlenia polskiej litery „ł” wykorzystaj generator znaków z pamięci CGRAM. W celu wyświetlenia całego napisu wykorzystaj przesuwanie wyświetlacza.
5. Zrealizuj animację napisu „Animacja testowa” w taki sposób, aby napis najpierw pojawiał się znak po znaku, a następnie był kasowany od ostatniego znaku do pierwszego. W trakcie animacji kursor ma być widoczny.

Zadanie 1

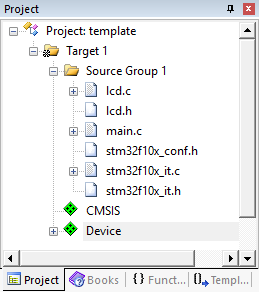
**Treść:**

Utwórz projekt, do którego należy dodać zawartość pliku cw5p1.zip. Uzupełnij program o brakujące parametry komend (plik lcd.h) oraz procedury inicjalizacji i wysyłania znaku do pamięci DDRAM (plik lcd.c). Zademonstruj działanie wyświetlacza wypisując napis np. „Hello World!”

**Realizacja:**

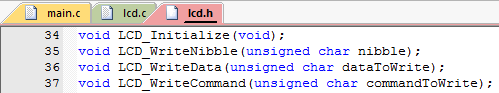
Najpierw podłączyliśmy porty Con13 do portów Con17 (D4 → PB0, D5 → PB1, D6 → PB2, D7 → PB3, E → PB4, RS → PB5, RW → GND), co było wstępnym przygotowanie płytki do tego i następnych zadań. Następnie otworzyliśmy projekt template:

* dodaliśmy zawartość pliku cw5p1.zip:



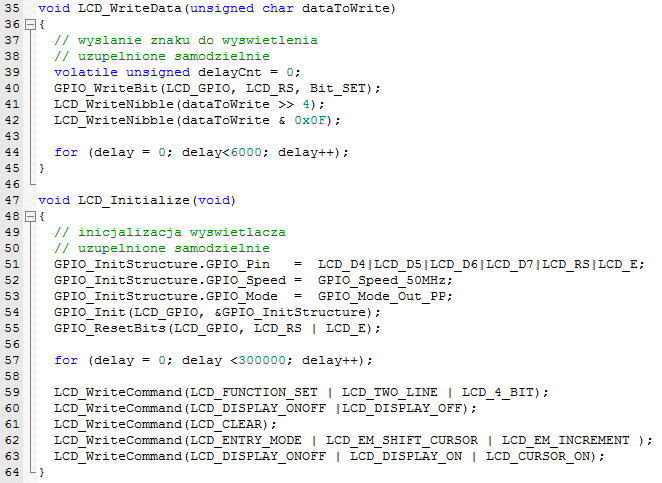
Rysunek 1

* uzupełniliśy program o brakujące parametry komend (plik lcd.h):

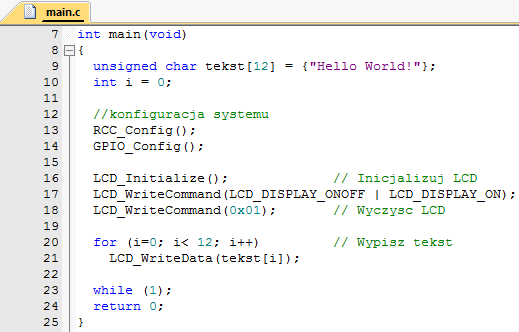


Rysunek 2

* uzupełniliśy procedury inicjalizacji i wysyłania znaku do pamięci DDRAM (plik lcd.c):



Rysunek 3



Rysunek 4

Zadamonstowaliśmy działanie wyświetlacza wypisując napis “Hello World!” - wszystko zadziałało.

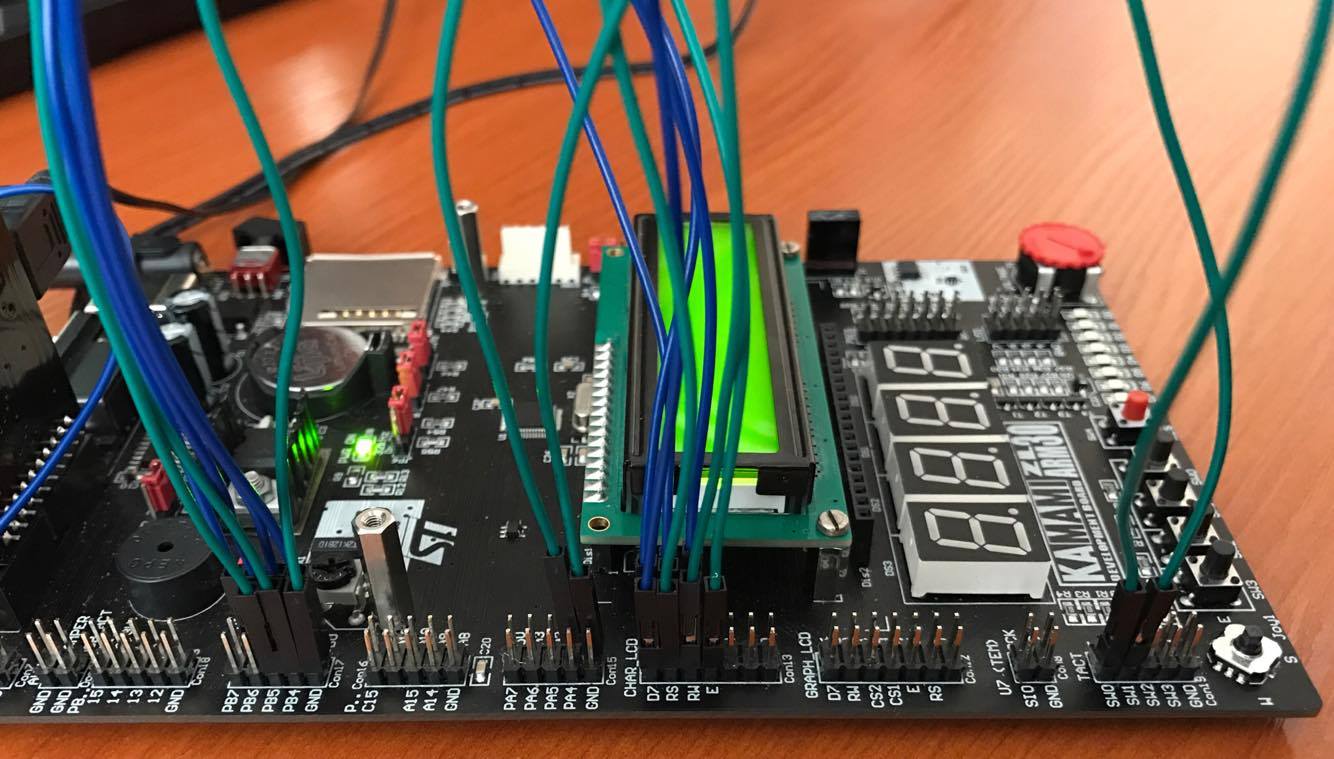
Zadanie 2

**Treść:**

Napisz program, który wypełni pamięć DDRAM znakami o kodach od 0x20 do 0x6F. Następnie każdorazowe wciśnięcie przycisku S1 ma przesunąć wyświetlacz w lewo, a przycisku S2 – w prawo.

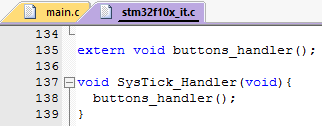
**Realizacja:**

Wykonanie tego zadania zaczęliśmy od podłączenia dodatkowo dwóch przycisków N → Con15 ( SW1 → PA0, SW2 → PA1:



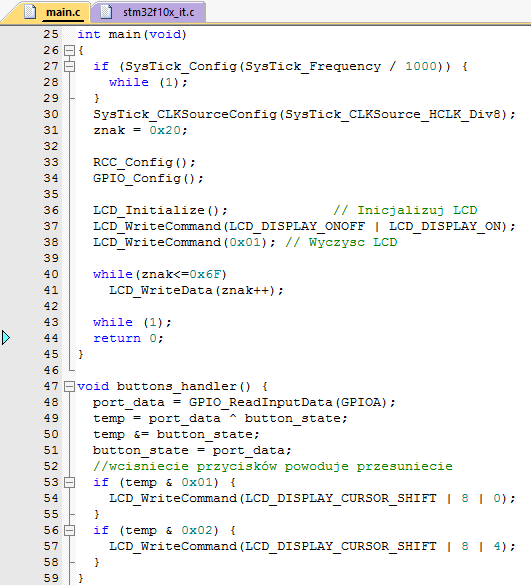
Rysunek 5

Obsługę przycisków zrealizowaliśmy na podstawie SysTick’a:



Rysunek 6

Realizacja buttons\_handler’a i kod z main(void):



Rysunek 7

Napisany program działa zgodznie z zadaniem i oczekiwaniami.

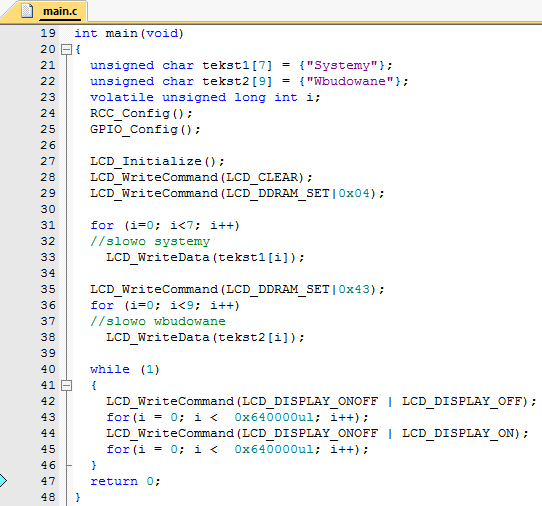
Zadanie 3

**Treść:**

Napisz na wyświetlaczu w pierwszym wierszu na środku słowo „Systemy”, a w drugim na środku słowo „Wbudowane”. Następnie spraw, aby cały wyświetlacz migał z okresem 1 sek.

**Realizacja:**

Wszyskie porty były już podłączone więc od razu zabraliśmy się za napisanie programu.



Rysunek 8

Napisany program działa zgodnie z zadaniem i oczekiwaniami mimo, że miganie nie wynosi dokładnie 1 sekundę.

Zadanie 4

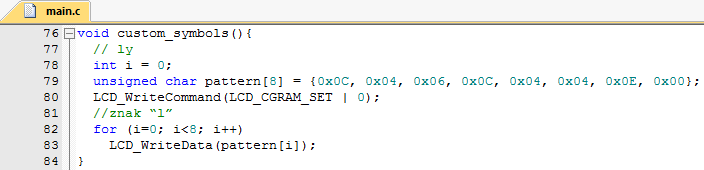
**Treść:**

Napisz program, który wyświetli na ekranie napis „Programowanie ARM jest łatwe”. Do wyświetlenia polskiej litery „ł” wykorzystaj generator znaków z pamięci CGRAM. W celu wyświetlenia całego napisu wykorzystaj przesuwanie wyświetlacza.

**Realizacja:**

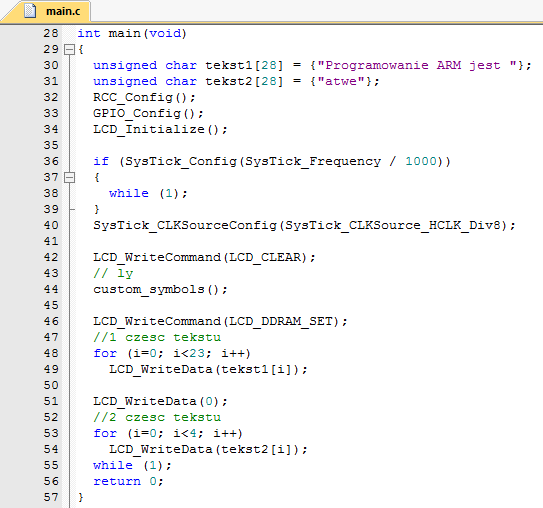
W tym zadaniu w celu przesuwania wyświetlacza wykorzytaliśmy dwa przyciski tak jak w zadaniu 2.

Kod programu wziązany z zdefniowaniem własnego znaku zamknęliśmy w dodatkowej fukcji void custom\_symbols(void):



Rysunek 9

Całą resztę kodu standardowo napisaliśmy w main(void):



Rysunek 10

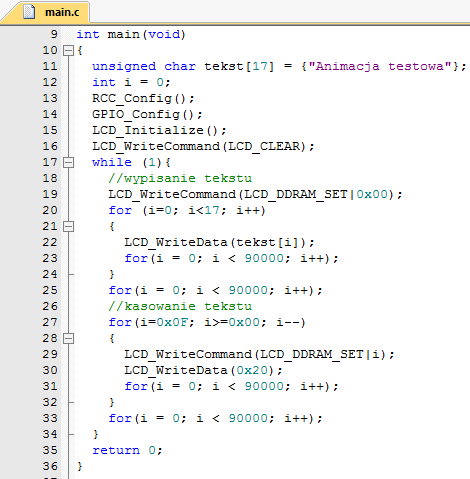
Zadanie 5

**Treść:**

Zrealizuj animację napisu „Animacja testowa” w taki sposób, aby napis najpierw pojawiał się znak po znaku, a następnie był kasowany od ostatniego znaku do pierwszego. W trakcie animacji kursor ma być widoczny.

**Realizacja:**

Całe zadanie zrealizowaliśmy w main(void):



Rysunek 11

Efekt wykonanego zadania tak nas zafascynował, że dopiero w domu zobaczyliśmy na nagranym filmiku, że zapomnieliśmy wyświetlać kursor: [link do filmiku](https://youtu.be/4G1tvhkrtjw). Mimo to napis wyświetlany jest w fajnej animacji.

**Kody źródłowe:**

**Zadanie 1:**

**Plik lcd.h**

#define LCD\_GPIO GPIOB

#define LCD\_D4 GPIO\_Pin\_0

#define LCD\_D5 GPIO\_Pin\_1

#define LCD\_D6 GPIO\_Pin\_2

#define LCD\_D7 GPIO\_Pin\_3

#define LCD\_E GPIO\_Pin\_4

#define LCD\_RS GPIO\_Pin\_5

#define LCD\_CLEAR 0x01

#define LCD\_HOME 0x02

#define LCD\_ENTRY\_MODE 0x04

#define LCD\_EM\_SHIFT\_CURSOR 0

#define LCD\_EM\_INCREMENT 2

#define LCD\_DISPLAY\_ONOFF 0x08

#define LCD\_DISPLAY\_OFF 0

#define LCD\_DISPLAY\_ON 4

#define LCD\_CURSOR\_ON 2

#define LCD\_CURSOR\_BLINK 1

#define LCD\_DISPLAY\_CURSOR\_SHIFT 0x10

#define LCD\_FUNCTION\_SET0x20

#define LCD\_FONT8 0

#define LCD\_TWO\_LINE 8

#define LCD\_4\_BIT 0

#define LCD\_CGRAM\_SET 0x40

#define LCD\_DDRAM\_SET 0x80

void LCD\_Initialize(void);

void LCD\_WriteData(unsigned char dataToWrite);

void LCD\_WriteCommand(unsigned char commandToWrite);

**Plik lcd.c**

#include "lcd.h"

#include "stm32f10x\_gpio.h"

GPIO\_InitTypeDef GPIO\_InitStructure;

// wyslij pólbajt na linie danych wyswietlacza

void LCD\_WriteNibble(unsigned char nibble)

{

volatile unsigned int delayCnt = 0;

GPIO\_WriteBit(LCD\_GPIO, LCD\_D4, (nibble & 0x01)); // ustaw bity na liniach

GPIO\_WriteBit(LCD\_GPIO, LCD\_D5, (nibble & 0x02)); // D4 – D7

GPIO\_WriteBit(LCD\_GPIO, LCD\_D6, (nibble & 0x04));

GPIO\_WriteBit(LCD\_GPIO, LCD\_D7, (nibble & 0x08));

GPIO\_WriteBit(LCD\_GPIO, LCD\_E, Bit\_SET); // ustaw wysoki poziom E

for(delayCnt = 0; delayCnt < 16; delayCnt++); // poczekaj troche…

GPIO\_WriteBit(LCD\_GPIO, LCD\_E, Bit\_RESET); // ustaw niski poziom E

}

// wyslij komende do wyswietlacza

void LCD\_WriteCommand(unsigned char commandToWrite)

{

volatile unsigned int delayCnt = 0;

GPIO\_WriteBit(LCD\_GPIO, LCD\_RS, Bit\_RESET); // RS = 0 - komenda

LCD\_WriteNibble(commandToWrite >> 4); // wyslij starszy pólbajt

LCD\_WriteNibble(commandToWrite & 0x0F); // wyslij mlodszy pólbajt

if (commandToWrite > 3) // w zaleznosci od komendy dobierz opóznienie

for(delayCnt = 0; delayCnt < 3000; delayCnt++);

else

for(delayCnt = 0; delayCnt < 150000; delayCnt++);

}

// wyslij znak do wyswietlenia

void LCD\_WriteData(unsigned char dataToWrite)

{

volatile unsigned delayCnt = 0;

GPIO\_WriteBit(LCD\_GPIO, LCD\_RS, Bit\_SET);

LCD\_WriteNibble(dataToWrite >> 4);

LCD\_WriteNibble(dataToWrite & 0x0F);

for (delayCnt = 0; delayCnt<6000; delayCnt++);

}

// inicjalizacja wyswietlacza

void LCD\_Initialize(void)

{

volatile unsigned char i = 0;

volatile unsigned int delayCnt = 0;

GPIO\_InitStructure.GPIO\_Pin = LCD\_D4|LCD\_D5|LCD\_D6|LCD\_D7|LCD\_RS|LCD\_E;

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

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

GPIO\_Init(LCD\_GPIO, &GPIO\_InitStructure);

GPIO\_ResetBits(LCD\_GPIO, LCD\_RS | LCD\_E);

for (delayCnt = 0; delayCnt <300000; delayCnt++);

LCD\_WriteCommand(LCD\_FUNCTION\_SET | LCD\_TWO\_LINE | LCD\_4\_BIT);

LCD\_WriteCommand(LCD\_DISPLAY\_ONOFF |LCD\_DISPLAY\_OFF);

LCD\_WriteCommand(LCD\_CLEAR);

LCD\_WriteCommand(LCD\_ENTRY\_MODE | LCD\_EM\_SHIFT\_CURSOR | LCD\_EM\_INCREMENT );

LCD\_WriteCommand(LCD\_DISPLAY\_ONOFF | LCD\_DISPLAY\_ON | LCD\_CURSOR\_ON);

}

**Plik main.c**

#include "stm32f10x.h"

#include "lcd.h"

void GPIO\_Config(void);

void RCC\_Config(void);

int main(void)

{

unsigned char tekst[12] = {"Hello World1"};

int i = 0;

RCC\_Config();

GPIO\_Config();

LCD\_Initialize(); // Inicjalizuj LCD

LCD\_WriteCommand(LCD\_DISPLAY\_ONOFF | LCD\_DISPLAY\_ON);

LCD\_WriteCommand(0x01); // Wyczysc LCD

for (i=0; i< 12; i++) // Wypisz tekst

LCD\_WriteData(tekst[i]);

while (1);

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) == 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 | RCC\_APB2Periph\_GPIOA | RCC\_APB2Periph\_AFIO, ENABLE);//wlacz taktowanie portu GPIO B

} else {

}

}

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 z konfiguracja poszczegolnych portow GPIO potrzebnych w programie\*/

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);

}

**Zadanie 2:**

**Pliki lcd.h i lcd.c** są takie samejak z zadaniu 1.

**Plik stm32f10x\_it.c:**Zmieniony fragment związany z obsługą handlera SysTick’a:

extern void buttons\_handler();

void SysTick\_Handler(void){

buttons\_handler();

}

**Plik main.c:**Początek main.c (pozostałe funckje nie zostały zmienione – więc są tak jak w zadaniu 1):

#include "stm32f10x.h"

#include "lcd.h"

#define SysTick\_Frequency 9000000

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

int znak;

int i = 0;

void GPIO\_Config(void);

void RCC\_Config(void);

int main(void)

{

if (SysTick\_Config(SysTick\_Frequency / 1000)) {

while (1);

}

SysTick\_CLKSourceConfig(SysTick\_CLKSource\_HCLK\_Div8);

znak = 0x20;

RCC\_Config();

GPIO\_Config();

LCD\_Initialize(); // Inicjalizuj LCD

LCD\_WriteCommand(LCD\_DISPLAY\_ONOFF | LCD\_DISPLAY\_ON);

LCD\_WriteCommand(0x01); // Wyczysc LCD

while(znak<=0x6F)

LCD\_WriteData(znak++);

while (1);

return 0;

}

void buttons\_handler() {

port\_data = GPIO\_ReadInputData(GPIOA);

temp = port\_data ^ button\_state;

temp &= button\_state;

button\_state = port\_data;

//wcisniecie przycisków powoduje przesuniecie

if (temp & 0x01) {

LCD\_WriteCommand(LCD\_DISPLAY\_CURSOR\_SHIFT | 8 | 0);

}

if (temp & 0x02) {

LCD\_WriteCommand(LCD\_DISPLAY\_CURSOR\_SHIFT | 8 | 4);

}

}

**Zadanie 3:**

**Pliki lcd.h i lcd.c** są takie samejak z zadaniu 1.

**Plik main.c:**  
Początek main.c (pozostałe funckje nie zostały zmienione – więc są tak jak w zadaniu 1):

#include "stm32f10x.h"

#include "lcd.h"

void GPIO\_Config(void);

void RCC\_Config(void);

int main(void)

{

unsigned char tekst1[7] = {"Systemy"};

unsigned char tekst2[9] = {"Wbudowane"};

volatile unsigned long int i;

RCC\_Config();

GPIO\_Config();

LCD\_Initialize();

LCD\_WriteCommand(LCD\_CLEAR);

LCD\_WriteCommand(LCD\_DDRAM\_SET|0x04);

for (i=0; i<7; i++)

//slowo systemy

LCD\_WriteData(tekst1[i]);

LCD\_WriteCommand(LCD\_DDRAM\_SET|0x43);

for (i=0; i<9; i++)

//slowo wbudowane

LCD\_WriteData(tekst2[i]);

while (1)

{

LCD\_WriteCommand(LCD\_DISPLAY\_ONOFF | LCD\_DISPLAY\_OFF);

for(i = 0; i < 0x640000ul; i++);

LCD\_WriteCommand(LCD\_DISPLAY\_ONOFF | LCD\_DISPLAY\_ON);

for(i = 0; i < 0x640000ul; i++);

}

return 0;

}

**Zadanie 4:**

**Pliki lcd.h i lcd.c** są takie samejak z zadaniu 1.

**Plik stm32f10x\_it.c** jest taki sam jak w zadaniu 2 (do obsługi SysTick’a).

**Plik main.c:**  
Początek pliku main.c (pozostałe funckje nie zostały zmienione – więc są tak jak w zadaniu 1):

#include "stm32f10x.h"

#include "lcd.h"

#define SysTick\_Frequency 9000000

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

int i = 0;

void GPIO\_Config(void);

void RCC\_Config(void);

void custom\_symbols(void);

int main(void)

{

unsigned char tekst1[28] = {"Programowanie ARM jest "};

unsigned char tekst2[28] = {"atwe"};

RCC\_Config();

GPIO\_Config();

LCD\_Initialize();

if (SysTick\_Config(SysTick\_Frequency / 1000))

{

while (1);

}

SysTick\_CLKSourceConfig(SysTick\_CLKSource\_HCLK\_Div8);

LCD\_WriteCommand(LCD\_CLEAR);

// ly

custom\_symbols();

LCD\_WriteCommand(LCD\_DDRAM\_SET);

//1 czesc tekstu

for (i=0; i<23; i++)

LCD\_WriteData(tekst1[i]);

LCD\_WriteData(0);

//2 czesc tekstu

for (i=0; i<4; i++)

LCD\_WriteData(tekst2[i]);

while (1);

return 0;

}

void check\_buttons() {

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

temp = port\_data ^ button\_state; // czy stan przycisków sie zmienil?

temp &= button\_state; // czy to byla zmiana z 1 na 0?

button\_state = port\_data; // zapamietaj nowy stan

//przesuwanie wyswietlacza

if (temp & 0x01) {

LCD\_WriteCommand(LCD\_DISPLAY\_CURSOR\_SHIFT | 8 | 0);

}

if (temp & 0x02) {

LCD\_WriteCommand(LCD\_DISPLAY\_CURSOR\_SHIFT | 8 | 4);

}

}

void custom\_symbols(){

// ly

int i = 0;

unsigned char pattern[8] = {0x0C, 0x04, 0x06, 0x0C, 0x04, 0x04, 0x0E, 0x00};

LCD\_WriteCommand(LCD\_CGRAM\_SET | 0);

//znak “l”

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

LCD\_WriteData(pattern[i]);

}

**Zadanie 5:**

**Pliki lcd.h i lcd.c** są takie samejak z zadaniu 1.

**Plik main.c:**  
Początek pliku main.c (pozostałe funckje nie zostały zmienione – więc są tak jak w zadaniu 1):

#include "stm32f10x.h"

#include "lcd.h"

void GPIO\_Config(void);

void RCC\_Config(void);

int main(void)

{

unsigned char tekst[17] = {"Animacja testowa"};

int i = 0;

RCC\_Config();

GPIO\_Config();

LCD\_Initialize();

LCD\_WriteCommand(LCD\_CLEAR);

while (1){

//wypisanie tekstu

LCD\_WriteCommand(LCD\_DDRAM\_SET|0x00);

for (i=0; i<17; i++)

{

LCD\_WriteData(tekst[i]);

for(i = 0; i < 90000; i++);

}

for(i = 0; i < 90000; i++);

//kasowanie tekstu

for(i=0x0F; i>=0x00; i--)

{

LCD\_WriteCommand(LCD\_DDRAM\_SET|i);

LCD\_WriteData(0x20);

for(i = 0; i < 90000; i++);

}

for(i = 0; i < 90000; i++);

}

return 0;

}