

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**

SW, semestr VI, 27-03-2017, Wydział Informatyki, Politechnika Białostocka

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). Zadeemonstruj 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

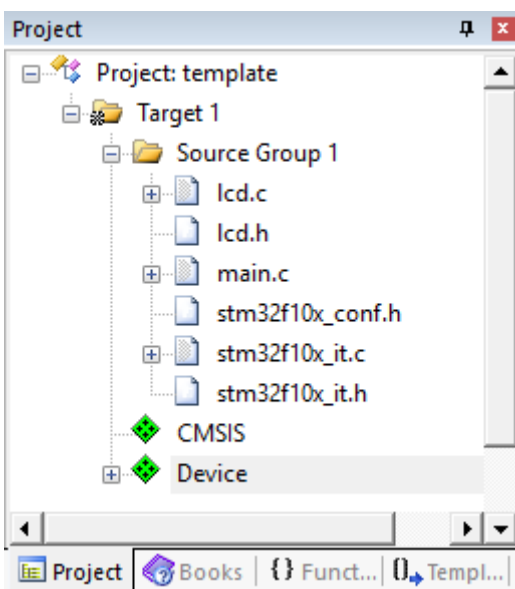
Treść:

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). Zadeemonstruj 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 przygotowaniem płytki do tego i następnych zadań. Następnie otworzyliśmy projekt template:

- dodaliśmy zawartość pliku cw5p1.zip:



Rysunek 1

- uzupełniliśmy program o brakujące parametry komend (plik lcd.h):

```
main.c  lcd.c  lcd.h
34 void LCD_Initialize(void);
35 void LCD_WriteNibble(unsigned char nibble);
36 void LCD_WriteData(unsigned char dataToWrite);
37 void LCD_WriteCommand(unsigned char commandToWrite);
```

Rysunek 2

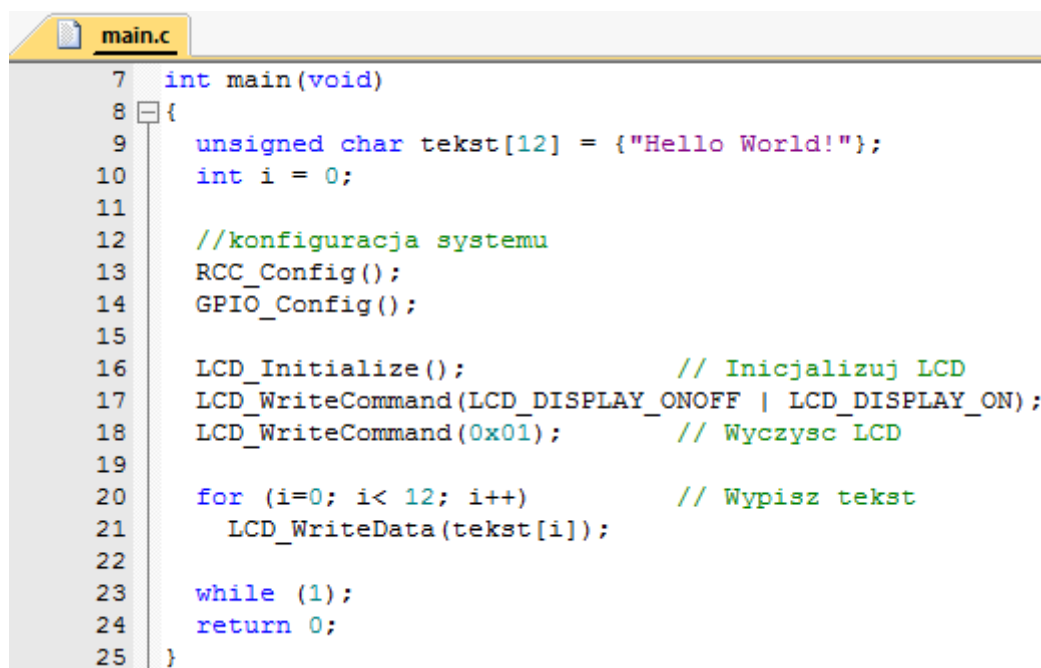
- uzupełniliśy procedury inicjalizacji i wysyłania znaku do pamięci DDRAM (plik lcd.c):

```

35 void LCD_WriteData(unsigned char dataToWrite)
36 {
37     // wysłanie znaku do wyświetlenia
38     // uzupełnione samodzielnie
39     volatile unsigned delayCnt = 0;
40     GPIO_WriteBit(LCD_GPIO, LCD_RS, Bit_SET);
41     LCD_WriteNibble(dataToWrite >> 4);
42     LCD_WriteNibble(dataToWrite & 0x0F);
43
44     for (delay = 0; delay<6000; delay++);
45 }
46
47 void LCD_Initialize(void)
48 {
49     // inicjalizacja wyświetlacza
50     // uzupełnione samodzielnie
51     GPIO_InitStructure.GPIO_Pin = LCD_D4|LCD_D5|LCD_D6|LCD_D7|LCD_RS|LCD_E;
52     GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
53     GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
54     GPIO_Init(LCD_GPIO, &GPIO_InitStructure);
55     GPIO_ResetBits(LCD_GPIO, LCD_RS | LCD_E);
56
57     for (delay = 0; delay <300000; delay++);
58
59     LCD_WriteCommand(LCD_FUNCTION_SET | LCD_TWO_LINE | LCD_4_BIT);
60     LCD_WriteCommand(LCD_DISPLAY_ONOFF | LCD_DISPLAY_OFF);
61     LCD_WriteCommand(LCD_CLEAR);
62     LCD_WriteCommand(LCD_ENTRY_MODE | LCD_EM_SHIFT_CURSOR | LCD_EM_INCREMENT );
63     LCD_WriteCommand(LCD_DISPLAY_ONOFF | LCD_DISPLAY_ON | LCD_CURSOR_ON);
64 }

```

Rysunek 3



```

main.c
7 int main(void)
8 {
9     unsigned char tekst[12] = {"Hello World!"};
10    int i = 0;
11
12    //konfiguracja systemu
13    RCC_Config();
14    GPIO_Config();
15
16    LCD_Initialize();           // Inicjalizuj LCD
17    LCD_WriteCommand(LCD_DISPLAY_ONOFF | LCD_DISPLAY_ON);
18    LCD_WriteCommand(0x01);    // Wyczyść LCD
19
20    for (i=0; i< 12; i++)      // Wypisz tekst
21        LCD_WriteData(tekst[i]);
22
23    while (1);
24    return 0;
25 }

```

Rysunek 4

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

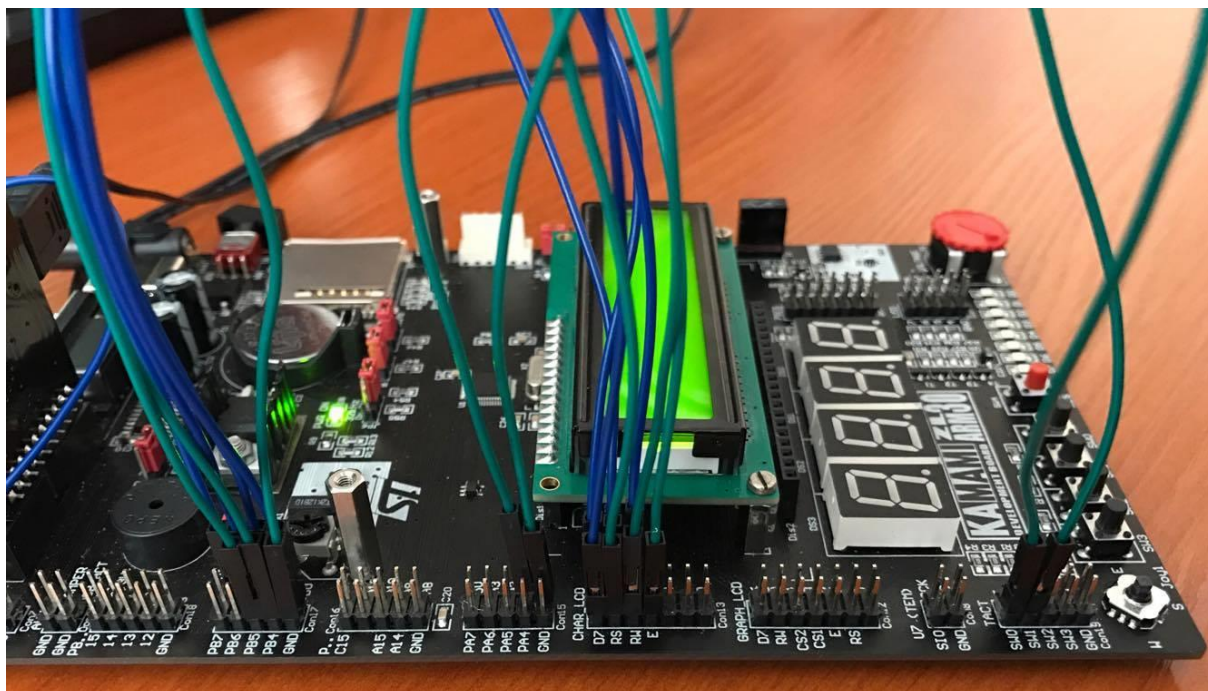
Zadanie 2

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:



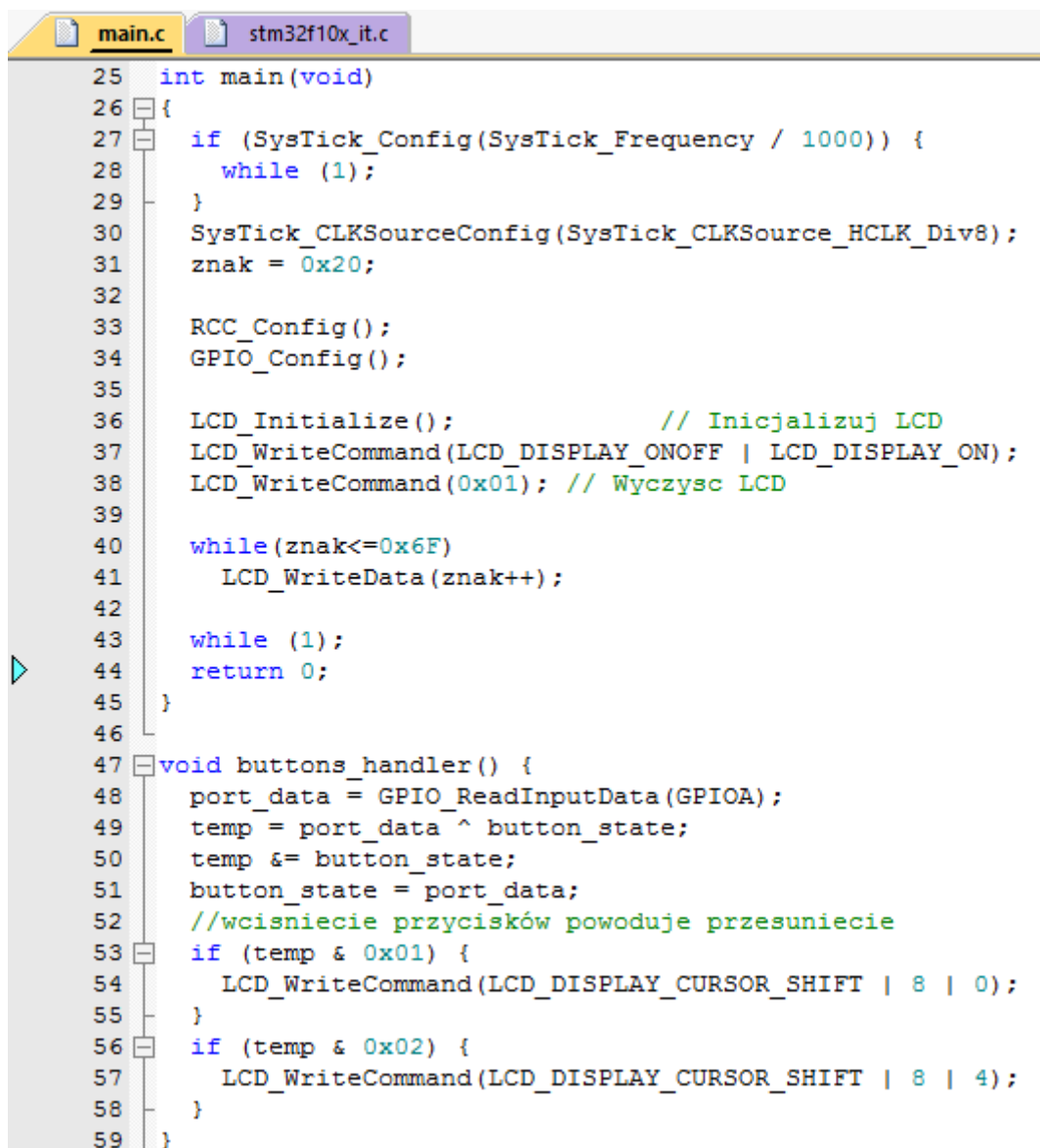
Rysunek 5

Obsługę przycisków zrealizowaliśmy na podstawie SysTick'a:

```
main.c | stm32f10x_it.c
134 |
135 | extern void buttons_handler();
136 |
137 | void SysTick_Handler(void) {
138 |     buttons_handler();
139 | }
```

Rysunek 6

Realizacja buttons_handler'a i kod z main(void):



```
25 int main(void)
26 {
27     if (SysTick_Config(SysTick_Frequency / 1000)) {
28         while (1);
29     }
30     SysTick_CLKSourceConfig(SysTick_CLKSource_HCLK_Div8);
31     znak = 0x20;
32
33     RCC_Config();
34     GPIO_Config();
35
36     LCD_Initialize(); // Inicjalizuj LCD
37     LCD_WriteCommand(LCD_DISPLAY_ONOFF | LCD_DISPLAY_ON);
38     LCD_WriteCommand(0x01); // Wyczyść LCD
39
40     while (znak <= 0x6F)
41         LCD_WriteData(znak++);
42
43     while (1);
44     return 0;
45 }
46
47 void buttons_handler() {
48     port_data = GPIO_ReadInputData(GPIOA);
49     temp = port_data ^ button_state;
50     temp &= button_state;
51     button_state = port_data;
52     //wcisniecie przycisków powoduje przesuniecie
53     if (temp & 0x01) {
54         LCD_WriteCommand(LCD_DISPLAY_CURSOR_SHIFT | 8 | 0);
55     }
56     if (temp & 0x02) {
57         LCD_WriteCommand(LCD_DISPLAY_CURSOR_SHIFT | 8 | 4);
58     }
59 }
```

Rysunek 7

Napisany program działa zgodnie z zadaniem i oczekiwaniami.

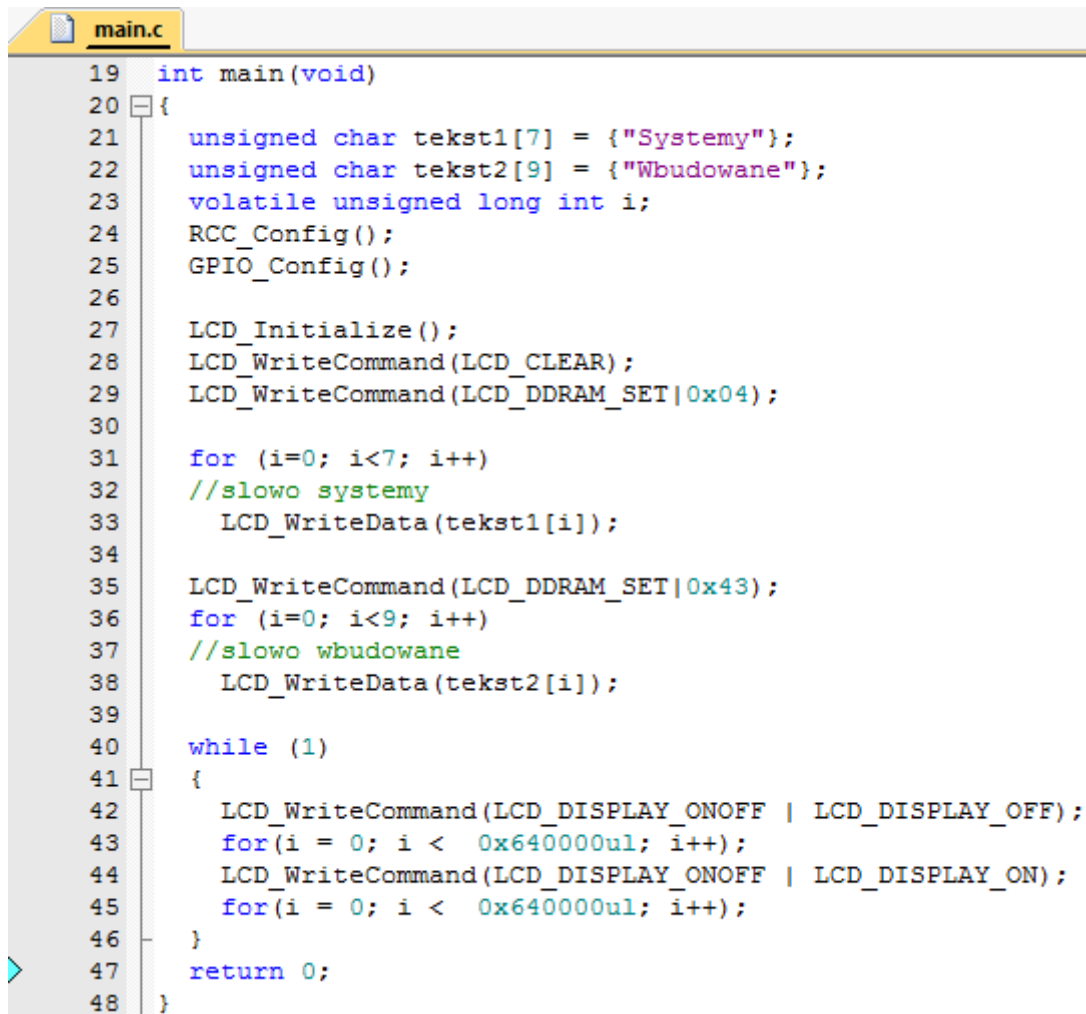
Zadanie 3

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:

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



```
19 int main(void)
20 {
21     unsigned char tekst1[7] = {"Systemy"};
22     unsigned char tekst2[9] = {"Wbudowane"};
23     volatile unsigned long int i;
24     RCC_Config();
25     GPIO_Config();
26
27     LCD_Initialize();
28     LCD_WriteCommand(LCD_CLEAR);
29     LCD_WriteCommand(LCD_DDRAM_SET|0x04);
30
31     for (i=0; i<7; i++)
32         //slowo systemy
33         LCD_WriteData(tekst1[i]);
34
35     LCD_WriteCommand(LCD_DDRAM_SET|0x43);
36     for (i=0; i<9; i++)
37         //slowo wbudowane
38         LCD_WriteData(tekst2[i]);
39
40     while (1)
41     {
42         LCD_WriteCommand(LCD_DISPLAY_ONOFF | LCD_DISPLAY_OFF);
43         for(i = 0; i < 0x640000ul; i++);
44         LCD_WriteCommand(LCD_DISPLAY_ONOFF | LCD_DISPLAY_ON);
45         for(i = 0; i < 0x640000ul; i++);
46     }
47     return 0;
48 }
```

Rysunek 8

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

Zadanie 4

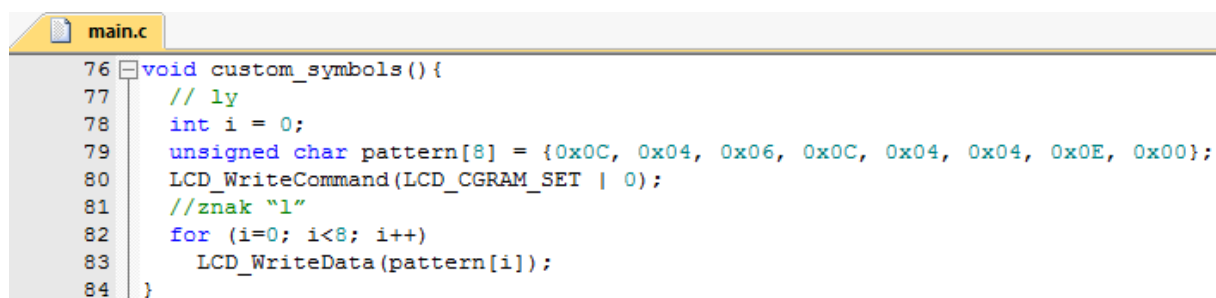
Treść:

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 wykorzystaliśmy dwa przyciski tak jak w zadaniu 2.

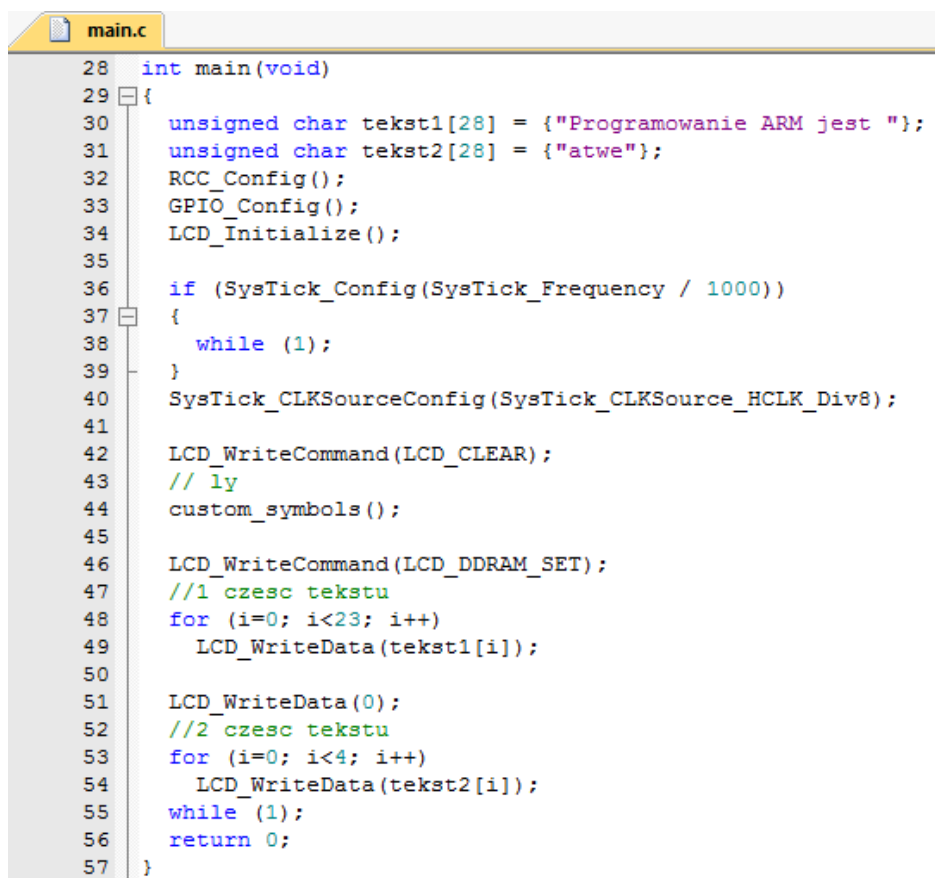
Kod programu wzięty z zdefiniowaniem własnego znaku zamknęliśmy w dodatkowej funkcji void custom_symbols(void):



```
76 void custom_symbols() {
77     // 1y
78     int i = 0;
79     unsigned char pattern[8] = {0x0C, 0x04, 0x06, 0x0C, 0x04, 0x04, 0x0E, 0x00};
80     LCD_WriteCommand(LCD_CGRAM_SET | 0);
81     //znak "ł"
82     for (i=0; i<8; i++)
83         LCD_WriteData(pattern[i]);
84 }
```

Rysunek 9

Całą resztę kodu standardowo napisaliśmy w main(void):



```
28 int main(void)
29 {
30     unsigned char tekst1[28] = {"Programowanie ARM jest "};
31     unsigned char tekst2[28] = {"atwe"};
32     RCC_Config();
33     GPIO_Config();
34     LCD_Initialize();
35
36     if (SysTick_Config(SysTick_Frequency / 1000))
37     {
38         while (1);
39     }
40     SysTick_CLKSourceConfig(SysTick_CLKSource_HCLK_Div8);
41
42     LCD_WriteCommand(LCD_CLEAR);
43     // 1y
44     custom_symbols();
45
46     LCD_WriteCommand(LCD_DDRAM_SET);
47     //1 czesc tekstu
48     for (i=0; i<23; i++)
49         LCD_WriteData(tekst1[i]);
50
51     LCD_WriteData(0);
52     //2 czesc tekstu
53     for (i=0; i<4; i++)
54         LCD_WriteData(tekst2[i]);
55     while (1);
56     return 0;
57 }
```

Rysunek 10

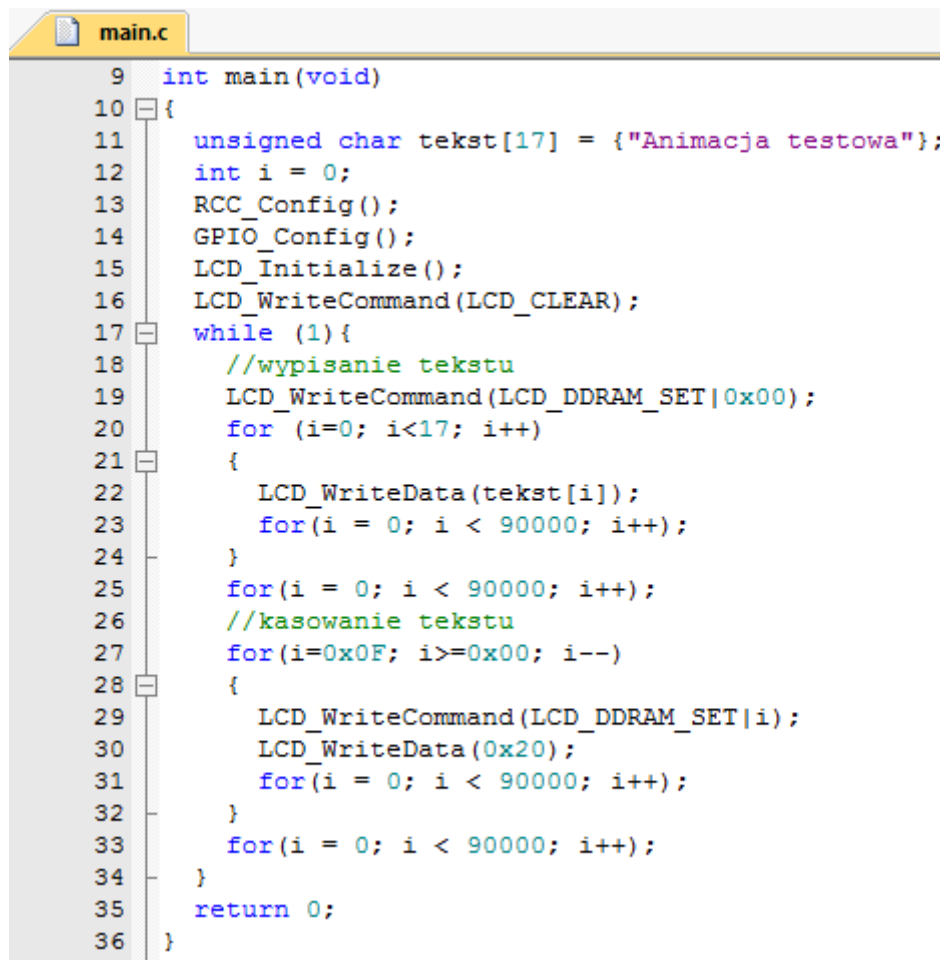
Zadanie 5

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):



```
9  int main(void)
10 {
11     unsigned char tekst[17] = {"Animacja testowa"};
12     int i = 0;
13     RCC_Config();
14     GPIO_Config();
15     LCD_Initialize();
16     LCD_WriteCommand(LCD_CLEAR);
17     while (1){
18         //wypisanie tekstu
19         LCD_WriteCommand(LCD_DDRAM_SET|0x00);
20         for (i=0; i<17; i++)
21         {
22             LCD_WriteData(tekst[i]);
23             for(i = 0; i < 90000; i++);
24         }
25         for(i = 0; i < 90000; i++);
26         //kasowanie tekstu
27         for(i=0x0F; i>=0x00; i--)
28         {
29             LCD_WriteCommand(LCD_DDRAM_SET|i);
30             LCD_WriteData(0x20);
31             for(i = 0; i < 90000; i++);
32         }
33         for(i = 0; i < 90000; i++);
34     }
35     return 0;
36 }
```

Rysunek 11

Efekt wykonanego zadania tak nas zafascynował, że dopiero w domu zobaczyliśmy na nagrany filmiku, że zapomnieliśmy wyświetlać kursor: [link do filmiku](#). 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 pólбайт na linie danych wyświetlacza
```

```

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
półbajt
    LCD_WriteNibble(commandToWrite & 0x0F);          // wyslij mlodszy
półbajt

    if (commandToWrite > 3)                          // w zaleznosci od komendy dobierz
opóźnienie
        for(delayCnt = 0; delayCnt < 3000; delayCnt++);
    else
        for(delayCnt = 0; delayCnt < 15000; delayCnt++);
}

// wyslij znak do wyswietlenia
void LCD_WriteData(unsigned char dataToWrite)
{
    volatile unsigned int 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;
    SW, semestr VI, 27-03-2017, Wydział Informatyki, Politechnika Białostocka

```

```

        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); // Wyczyszc 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);        //Wlaczanie
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 same jak 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 funkcje 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); // Wyczyszc 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 same jak z zadaniu 1.

Plik main.c:

Początek main.c (pozostałe funkcje 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 same jak 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 funkcje 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 same jak z zadaniu 1.

Plik `main.c`:

Początek pliku `main.c` (pozostałe funkcje 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;
}
```