SPRAWOZDANIE

SYSTEMY WBUDOWANE

Obsługa układów peryferyjnych.
Wyświetlacz graficzny.

IMIĘ I NAZWISKO: Jacek Wójcik

NUMER ĆWICZENIA: 4

Grupa laboratoryjna: 10

Data wykonania ćwiczenia: 3.11.2021

Spis treści

1. Zadanie 1 - kod

1.1 main.c

```c
#include "lcd.h"                    // Dołączam moją bibliotekę

int main(void)
{
    LCD_INIT();                     // Inicjalizuję wyświetlacz
    char *imie = "Jacek Wojcik";    // Zapisuję swoje imię w zmiennej
    LCD_WRITE(imie);                // Wyświetlam imię na wyświetlaczu
    while (1)
    {
        asm volatile("nop");        // Wykonuję niekończoną pętlę przez resztę czasu
    }
}

```

1.2 lcd.h

```c
#ifndef LCD_H_
#define LCD_H_

#define F_CPU 1000000               // Ustawiam częstotliwość na zgodną z zestawem w laboratorium

#include <avr/io.h>                 // Dołączam biblioteki do obsługi portów i opóźnień
#include <util/delay.h>

void LCD_CLEAR();                   // Funkcja czyszcząca cały wyświetlacz

void LCD_INIT();                    // Funkcja inicjalizująca wyświetlacz

void LCD_WRITE(char *str);          // Funkcja wypisująca dane na wyświetlacz

void LCD_XY(int x, int y);          // Funkcja ustawiająca kursor na danej pozycji na wyświetlaczu

void LCD_CLEAR_XY(int x, int y);    // Funkcja czyszcząca wyświetlacz od danej pozycji

#endif
```

1.3 lcd.c

```c
#include "lcd.h"

void sendHalfByte(char data)
{
    PORTA |= 0x02;                              // Wysyłam flagę ENABLE
    PORTA = (PORTA & 0x0F) | (data & 0xF0);     // Wysyłam pół bajta
    PORTA &= 0xFD;                              // Usuwam flagę ENABLE
}

void sendByte(char data)
{
    sendHalfByte(data);                         // Wysyłam górną połowę bajtu
    _delay_ms(2);                               // Opóźnienie w celu wykrycia stanu "0" na pinie ENABLE
    sendHalfByte(data << 4);                    // Wysyłam dolną połowę bajtu
}

void sendCommand(char data)
{
    sendByte(data);                             // Wysyłam komendę
    _delay_ms(5);                               // Czekam na przetworzenie komendy
}

void sendChar(char data)
{
    sendByte(data);                             // Wysyłam znak
    _delay_ms(2);                               // Czekam na wypisanie znaku
}

void LCD_CLEAR()
{
    PORTA &= 0xFE;                              // Przełączam LCD w tryb wprowadzania komend
    _delay_ms(2);                               // Opóźnienie w celu wykrycia stanu "0" na pinie RS
    sendCommand(0x01);                          // Czyszczę wyświetlacz
    PORTA |= 0x01;                              // Przełączam LCD w tryb wprowadzania danych
}

void LCD_INIT()
{
    _delay_ms(15);                              // Czekam aż wyświetlacz LCD zostanie zainicjalizowany

    DDRA |= 0xF3;                               // Ustawiam część linii A

    // Inicjalizacja standardowymi bajtami
    sendHalfByte(0x30);
    _delay_ms(5);                               // Czekam zgodnie z dokumentacją
    sendHalfByte(0x30);
    _delay_ms(1);
    sendCommand(0x32);

```

```
49
50      // Inicjalizacja ustawień wyświetlacza
51      sendCommand(0x28);                  // Ustawiam tryb 2 linii i znaków 5x8
52      sendCommand(0x08);                  // Wyłączam wyświetlacz zgodnie z dokumentacją
53      sendCommand(0x01);                  // Czyszczę wyświetlacz
54      sendCommand(0x06);                  // Ustawiam kierunek wyprowadzania tekstu i sposób wyprowadzania na wyświetlacz
55      sendCommand(0x0F);                  // Włączam wyświetlacz i ustawiam znak na pozycji kursora na miganie
56      PORTA |= 0x01;                      // Przełączam LCD w tryb wprowadzania danych
57  }
58
59  void LCD_WRITE(char *str)
60  {
61      while(*str)                         // Dopóki nie dojdę to null terminatora wypisuję kolejne znaki
62      {
63          sendChar(*str++);               // Przesuwam się do kolejnego znaku po wypisaniu go na wyświetlaczu
64      }
65  }
66
67  void LCD_XY(int x, int y)
68  {
69      PORTA &= 0xFE;                      // Przełączam LCD w tryb wprowadzania komend
70      _delay_ms(2);                       // Opóźnienie w celu wykrycia stanu "0" na pinie RS
71      sendCommand(0x80 | x | y << 6);     // Ustawiam kursor na odpowowiedniej pozycji
72      PORTA |= 0x01;                      // Przełączam LCD w tryb wprowadzania danych
73  }
74
75  void LCD_CLEAR_XY(int x, int y)
76  {
77      LCD_XY(x,y);                        // Ustawiam kursor na pozycji od której chcę czyścić
78      if(y == 0)                          // Sprawdzam czy to pierwsza linia
79      {
80          while(x++ < 16)                 // Dopóki nie dojdę do końca linii
81          {
82              LCD_WRITE(" ");             // Wypisuję pusty znak na wyjście wyświetlacza
83          }
84          x = 0;                          // Ustawiam pierwszą składową adresu na początek linii
85          y++;                            // Ustawiam drugą składową adresu na drugą linię
86          LCD_XY(x,y);                    // Ustawiam kursor na podany adres -> początek drugiej linii
87      }
88      while(x++ < 16)                     // Dopóki nie dojdę do końca linii
89      {
90          LCD_WRITE(" ");                 // Wypisuję pusty znak na wyjście wyświetlacza
91      }
92  }
93
```

2. Zadanie 1 - opis

Zadanie 1 polegało na napisaniu biblioteki do podstawowej obsługi ekranu LCD a następnie wyświetlenie swojego imienia i nazwiska za jej pomocą. Zrealizowałem je poprzez wystawienie następującego API w mojej bibliotece:

Prototyp funkcji	**Argumenty**	**Opis**
LCD_INIT()	Brak	Funkcja inicjalizuje wyświetlacz poprzez: ● zaczekanie na wewnętrzne zainicjalizowanie wyświetlacza ● ustawienie portów mikrokontrolera podłączonych do wyświetlacza jako wyjście ● wysłanie standardowego zestawu instrukcji z dokumentacji ustawiającego m.in. tryb wyświetlania, pozycję kursora, wielkość znaków ● przełączenie wyświetlacza w tryb wprowadzania danych
LCD_WRITE (char * str)	● **char *str** - ciąg do wyświetlenia	Funkcja wyświetla ciąg znaków zawarty w argumencie **str** w formie pętli, która bierze po kolei każdy znak z ciągu i przesyła go do wyświetlacza.
LCD_XY (int x, int y)	● **int x** - numer znaku ● **int y** - numer linii	Funkcja ustawia kursor na pozycji wskazanej przez argumenty **x** i **y**. Dzieje się to poprzez wysłanie kodu komendy, który powstaje poprzez selektywne zsumowanie kodu **0x80** (sygnał, że chodzi o ustawienie kursora), argumentu **x** (numer znaku) i argumentu **y** przesuniętego bitowo w lewo o 6 (ponieważ bit 6 odpowiada za numer linii).
LCD_CLEAR()	Brak	Funkcja czyści cały ekran wyświetlacza poprzez ustawienie go w tryb przyjmowania poleceń, wysłanie polecenia wyczyszczenia wyświetlacza a następnie ponownego przełączenia go w tryb wprowadzania danych.
LCD_CLEAR_XY()	● **int x** - numer znaku ● **int y** - numer linii	Funkcja czyści ekran od wskazanego miejsca. Dzieje się to poprzez podmianę wszystkich znaków w danej linii (oraz każdej następnej) na pusty znak.

Dodatkowo w celu uproszczenia kodu napisałem następujące funkcje

- **sendHalfByte(char data)** - funkcja za pomocą PORTA wysyła pół bajta informacji do wyświetlacza.
- **sendByte(char data)** - funkcja za pomocą **sendHalfByte** przesyła cały bajt informacji w dwóch porcjach oddzielonych opóźnieniem w celu poprawnego odebrania ich przez wyświetlacz
- **sendCommand(char data)** - funkcja wysyła do wyświetlacza komendę funkcją **sendByte** i czeka 5ms w celu umożliwienia przetworzenia komendy
- **sendChar(char data)** - funkcja wysyła do wyświetlacza jeden znak i czeka 2ms w celu umożliwienia wyświetlenia znaku.

Ponieważ kable łączące mikrokontroler i ekran są długie i mają luźne połączenia, musiałem wielokrotnie zwiększyć każde opóźnienie względem wartości w dokumentacji.

3. Wnioski

Podczas tych laboratoriów napisałem własną bibliotekę do obsługi wyświetlacza LCD. Wykonanie tego zadania nauczyło mnie w jaki sposób komunikować się z innymi urządzeniami przez porty mikrokontrolera.

Pierwszym problemem, jaki napotkałem był fakt, że kable goldpinowe używane w zestawie w laboratorium posiadają parametry niezgodne z dokumentacją wyświetlacza, która zakłada lutowane połączenia pomiędzy mikrokontrolerem a wyświetlaczem a nie za pomocą goldpinów. W efekcie kable potrzebują znacznie większego (od 2 do 5 razy) czasu w celu ustabilizowania się sygnału tak, żeby został poprawnie odczytany przez urządzenie po drugiej stronie.

Na używałem początku opóźnień zgodnych z dokumentacją, co skończyło się brakiem odpowiedzi od wyświetlacza z powodu niepoprawnego odczytania instrukcji inicjalizujących. Rozwiązanie tego problemu zajęło mi kilkanaście minut, jako że najpierw szukałem błędów w logice mojego programu a nie w czasach opóźnień. Prawidłowym sposobem na skompensowanie tego problemu jest wydłużenie czasów opóźnień do czasu od 2ms do 5ms.

Wyciągnąłem z tego następujący wniosek: **zawsze należy uwzględniać sposób połączenia mikrokontrolera i układu peryferyjnego, jaki jest opisany w dokumentacji i jaki jest użyty w faktycznym układzie i dostosować program w celu korekty opóźnień umożliwiających stabilizację sygnału.**

Następnym problemem, na jaki napotkałem było wyczyszczenie ekranu mikrokontrolera od danego punktu. Mikrokontroler może przechowywać w pamięci DDRAM do 80 znaków, a wyświetlane jest tylko 28, pozostałe mogą zostać użyte jako generalna pamięć RAM[1]. Podczas czyszczenia ekranu wyświetlacza musiałem pamiętać, żeby, jeżeli np. czyszczę od 5 pozycji na linii górnej, wyczyścić (zastąpić pustymi znakami) tylko miejsca od 5 do 15 (numeruję miejsca od 0), a następnie “zawinąć” czyszczenie przechodząc do nowej linii (czyli przeskakując do adresu 0x40) i tam wyczyścić znaki od znaku 0 do 15. Dzięki temu jednocześnie czyszczę ekran i nie usuwam pozostałej zawartości pamięci DDRAM. Moje wnioski z tego problemu przedstawia poniższa tabela:

Adres pamięci DDRAM	**Opis**
0x00 - 0x0F	Komórki z tych adresów są wyświetlane w pierwszej linii na ekranie.
0x10 - 0x3F	Komórki z tych adresów mogą zostać użyte jako ogólna pamięć RAM.
0x40 - 0x4F	Komórki z tych adresów są wyświetlane w drugiej linii na ekranie.

Należy pamiętać, że zawartość pamięci DDRAM może zostać przesunięta (lub może zostać zmieniony tryb wyświetlania) co spowoduje zmianę adresów komórek pamięci, których zawartość jest wyświetlana na ekranie. Jednak jako że biblioteka nie zakłada przesuwania ani innych trybów wyświetlania, można założyć, że są to stałe wartości.

---

[1] Dokumentację wyświetlacza wziąłem z strony https://www.sparkfun.com/datasheets/LCD/HD44780.pdf
Informację odnośnie wielkości pamięci RAM wziąłem z strony 10, rozkład pamięci DDRAM z strony 12.