Ćwiczenie 5: Wyświetlacz alfanumeryczny LCD

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.

Wprowadzenie do ćwiczeń

Dokumentacja sterownika wyświetlacza alfanumerycznego HD44780 znajduje się dziale „Do pobrania” w sekcji „Materiały dodatkowe”.

Interfejs komunikacyjny składa się z następujących linii:

– D0-D7 – dwukierunkowa magistrala danych; w przypadku pracy 8-bitowej wykorzystywane są wszystkie sygnały, dla 4-bitowej starszy półbajt (D4-D7);

– RS (Register Select) – sygnał wyboru odbiorcy przesyłanych danych: gdy RS = 1 dane przesyłane są do pamięci DDRAM lub CGRAM, dla RS = 0 dane trafiają do rejestru konfiguracyjnego sterownika;

– R/W (Read/Write) – sygnał wyboru kierunku transmisji danych; dla R/W = 1 dane są odczytywane z wyświetlacza, gdy R/W = 0 dane są przesyłane do wyświetlacza;

– E (Enable) – sygnał strobujący magistrali danych; wartość E = 1 oznacza, że na magistrali danych są dane gotowe do odebrania.

Do poprawnej pracy wystarczy 6 linii: D4-D7 (interfejs 4-bitowy), RS oraz E. Sygnał R/W jest na stałe podłączony do logicznego „0”. W takiej konfiguracji nie jest możliwe odczytywanie danych z wyświetlacza, a co za tym idzie sprawdzenie, czy wyświetlacz jest gotowy do przyjmowania kolejnych danych. Dlatego wszystkie operacje na wyświetlaczu wymagają opóźnień zgodnych z informacjami zawartymi w dokumentacji wyświetlacza.

Wysyłanie danych do wyświetlacza wymaga ustawienia odpowiedniej wartości na linii RS, a następnie w dwóch cyklach przesłania bajtu. W pierwszym cyklu wysyłany jest starszy półbajt, a w drugim – młodszy. W każdym z cykli należy najpierw wystawić półbajt na linie danych, po czym ustawić wartość „1” na linii E, a po odczekaniu pewnego czasu (wartość z dokumentacji) linię E ustawić na „0”. Po wysłaniu bajtu należy odczekać, aż sterownik wyświetlacza zakończy wewnętrzną operację (ok. 40 µs, dokładne wartości w dokumentacji – dla niektórych komend czas oczekiwania powinien być dłuższy).

Procedura inicjalizacji wyświetlacza polega na wykonaniu następującej sekwencji czynności:

– odczekanie min. 15 ms;

– wysłanie komendy „Function Set” dla 8-bitowego interfejsu;

– odczekanie min. 4.1 ms;

– ponowne wysłanie komendy „Function Set” dla 8-bitowego interfejsu;

– odczekanie min. 100 µs;

– ponowne wysłanie komendy „Function Set” dla 8-bitowego interfejsu;

– odczekanie min. 40 µs;

– wysłanie komendy „Function Set” dla 4-bitowego interfejsu; w tym miejscu interfejs jest nadal 8-bitowy, więc wysyłany jest tylko starszy półbajt;
– odczekanie min. 40 µs;
– wysłanie komendy „Function Set” dla 4-bitowego interfejsu z jednoczesnym ustawieniem liczby wierszy i fontu;
– odczekanie min. 40 µs;
– wysłanie komendy „Display Off”;
– odczekanie min. 40 µs;
– wysłanie komendy „Display Clear”;
– odczekanie min. 40 µs;
– wysłanie komendy „Entry Mode Set” z odpowiednimi parametrami;
– odczekanie min. 40 µs.

Pamięć DDRAM służy do przechowywania kodów ASCII znaków wyświetlanych przez wyświetlacz. Organizacja pamięci zależy od budowy wyświetlacza: liczby wierszy i kolumn. Wyświetlacz zamontowany w zestawie ZL4PIC ma organizację 2x16 (dwa wiersze po 16 znaków). Domyślnie adres pierwszego wiersza wynosi 0x00, drugiego 0x40 (w przypadku pracy dwuwierszowej). Adresy zmieniają się po wykonaniu komendy „Display Shift” (więcej informacji w dokumentacji).

Pamięć CGRAM przechowuje mapy bitowe znaków definiowanych przez użytkownika. Istnieje możliwość zdefiniowania 8 własnych znaków dostępnych pod kodami 0-7.

Poniżej przedstawiono wykaz komend sterownika wyświetlacza.

Komenda	**Kod**								**Opis**
	D7	D6	D5	D4	D3	D2	D1	D0	
Display Clear	0	0	0	0	0	0	0	1	Kasowanie wyświetlacza
Return Home	0	0	0	0	0	0	1	0	Powrót do początku (również zerowanie przesunięcia)
Entry Mode Set	0	0	0	0	0	1	I/D	S	Tryb pracy
Display on/off	0	0	0	0	1	D	C	B	Włączanie/wyłączanie wyświetlacza
Cursor/Display Shift	0	0	0	1	S/C	R/L	–	–	Przesuwanie kursora/wyświetlacza
Function Set	0	0	1	DL	N	F	–	–	Funkcje podstawowe
Set CGRAM	0	1	Adres CGRAM						Ustawienie adresu CGRAM
Set DDRAM	1	Adres DDRAM							Ustawienie adresu DDRAM

I/D – zwiększanie (1) lub zmniejszanie (0) adresów po wpisaniu danych do pamięci;

S – włączanie (1) lub wyłączanie (0) przesuwania wyświetlacza;

D – włączanie (1) lub wyłączanie (0) wyświetlacza;

C – włączanie (1) lub wyłączanie (0) kursora;

B – włączanie (1) lub wyłączanie (0) migotania znaku na pozycji kursora;

S/C – przesuwanie wyświetlacza (1) lub kursora (0);

R/L – przesuwanie w prawo (1) lub w lewo (0);

DL – interfejs cztero (0) lub ośmio (1) bitowy;

N – wyświetlanie jednego (0) lub dwóch (1) wierszy;

F – 8 (0) lub 10 (1) linii wyświetlanego znaku.

Komenda *Display Clear* (kod 0x01) powoduje wyczyszczenie wyświetlacza (cała pamięć DDRAM jest wypełniana kodem spacji – 0x20), powrót kursora pod adres 0x00 oraz przywraca wyświetlanie od adresu 0x00 (likwiduje przesunięcie realizowane komendą *Display Shift*).

Komenda *Return Home* (kod 0x02) powoduje powrót kursora pod adres 0x00 oraz przywraca wyświetlanie od adresu 0x00 (likwiduje przesunięcie realizowane komendą *Display Shift*).

Komenda *Entry Mode Set* (kod 0x04) służy do ustawiania trybu pracy wyświetlacza. W zależności od wartości bitu I/D, każdorazowe wpisanie wartości do pamięci DDRAM powoduje zwiększenie (dla I/D = 1) lub zmniejszenie (dla I/D = 0) adresu. Dodatkowo istnieje możliwość włączenia przesuwania zawartości całego wyświetlacza poprzez ustawienie bitu S = 1.

Komenda *Display on/off* (kod 0x08) pozwala na włączenie (gdy D = 1) lub wyłączenie (gdy D = 0) wyświetlacza. Dodatkowo możliwe jest włączenie kursora (gdy C = 1) oraz migotania znaku znajdującego się na pozycji kursora (gdy B = 1).

Komenda *Cursor/Display Shift* (kod 0x10) pozwala na przesunięcie zawartości wyświetlacza (gdy S/C = 1) lub kursora (gdy S/C = 0) w prawo (gdy R/L = 1) lub w lewo (gdy R/L = 0) bez modyfikowania pamięci DDRAM.

Komenda *Set CGRAM* (kod 0x40) służy do ustawiania adresu pamięci CGRAM, pod który będą zapisywane dane. Pozwala na tworzenie własnych zestawów znaków. Po wykonaniu tej komendy każdorazowe wysłanie danej do wyświetlacza powoduje zapisanie jej w pamięci CGRAM. Po zapisaniu danej adres jest zwiększany lub zmniejszany w zależności od ostatnio ustawionej wartości bitu I/D komendy *Entry Mode Set*.

Komenda *Set DDRAM* (kod 0x80) służy do ustawiania adresu pamięci DDRAM, pod który będą zapisywane dane. Organizacja pamięci w trybie dwuwierszowym wygląda następująco (przy założeniu braku przesunięcia wyświetlania):

0x00	0x01	0x02	0x03	0x04	0x05	0x06	0x07	0x08	0x09	0x0A	0x0B	0x0C	0x0D	0x0E	0x0F
0x40	0x41	0x42	0x43	0x44	0x45	0x46	0x47	0x48	0x49	0x4A	0x4B	0x4C	0x4D	0x4E	0x4F

Rozmiar pamięci DDRAM wynosi 80 bajtów.

Pytania kontrolne

1. Jakie sygnały składają się na interfejs komunikacyjny z wyświetlaczem?
2. Jaka jest minimalna liczba sygnałów niezbędnych do poprawnego wysterowania wyświetlacza? Jakie to są sygnały?
3. Jak przebiega proces komunikacji z wyświetlaczem dla interfejsu 8-bitowego i 4-bitowego?
4. Jak przebiega proces inicjalizacji wyświetlacza?

5. Co to jest DDRAM?
6. Jaka jest organizacja pamięci DDRAM wyświetlacza?
7. Co to jest CGRAM?
8. Jak wyświetlić na ekranie znak?
9. Jakie znasz komendy wyświetlacza HD44780?

Przykłady:

1. **Elementy programu obsługi wyświetlacza LCD**

Program docelowo ma wyświetlać na ekranie wyświetlacza napis „Hello World!”. Program nie jest skończony. Wyprowadzenia złącza Con13 (CHAR_LCD) należy połączyć z wyprowadzeniami złącza Con17 (GPIOB) w następujący sposób: D4 → PB0, D5 → PB1, D6 → PB2, D7 → PB3, E → PB4, RS → PB5, RW → GND. Kody źródłowe programu – w pliku **cw5p1.zip**:

Plik *main.c* – program główny

```
#include "stm32f10x.h"
#include "lcd.h"

void GPIO_Config(void);
void RCC_Config(void);


int main(void)
{
      unsigned char tekst[12] = {"Hello World!"};
      int i = 0;

      RCC_Config();
      GPIO_Config();

      LCD_Initialize();                  // Inicjalizuj LCD
      LCD_WriteCommand(LCD_CLEAR);       // Wyczyść LCD

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

  while (1);
  return 0;
}
```

Plik *lcd.h* – definicje komend i portów

```
#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_SET                    0x20
      #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* – funkcje inicjalizacji, wysyłania komend i danych

```
#include "lcd.h"
#include "stm32f10x_gpio.h"

GPIO_InitTypeDef GPIO_InitStructure;

// wyślij półbajt 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
}

// wyślij komendę do wyświetlacza
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);          // wyślij starszy półbajt
      LCD_WriteNibble(commandToWrite & 0x0F);        // wyślij młodszy półbajt

      if (commandToWrite > 3)       // w zależności od komendy dobierz opóźnienie
            for(delayCnt = 0; delayCnt < 3000; delayCnt++);
      else
            for(delayCnt = 0; delayCnt < 150000; delayCnt++);
}
```

```
// wyślij znak do wyświetlenia
void LCD_WriteData(unsigned char dataToWrite)
{
      // uzupełnij samodzielnie
}

// inicjalizacja wyświetlacza
void LCD_Initialize(void)
{
      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);

      // uzupełnij samodzielnie
}
```

Zadania do samodzielnego 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.