Ćwiczenie 11: Obsługa sterownika wyświetlacza alfanumerycznego HD44780

Instrukcja laboratorium

Mariusz Chilmon <mariusz.chilmon@ctm.gdynia.pl>

2024-03-25

> Programming is the art of telling another human being what one wants the computer to do.
>
> — *Donald E. Knuth*

Cel ćwiczenia

Celem ćwiczenia jest zapoznanie się z:

- wymaganiami czasowymi w komunikacji z urządzeniami zewnętrznymi,
- wyświetlaniem tekstu na wyświetlaczach alfanumerycznych,
- prostymi operacjami na łańcuchach znaków.

Uruchomienie programu wyjściowego

1. Podłącz płytkę *LCD Keypad Shield* do *Arduino Uno*.
2. Wyświetlacz nie wyświetla żadnych znaków lub wyświetla przypadkowe symbole.

Używany wyświetlacz oparty jest na sterowniku Hitachi HD44780. Jest to sterownik wyświetlacza alfanumerycznego, czyli takiego, który jest w stanie wyświetlać tylko określony zestaw predefiniowanych znaków, np. liter, cyfr czy innych znaków z zakresu ASCII[1]. Uniemożliwia to lub co najmniej mocno ogranicza wyświetlanie dowolnych kształtów, ale odciąża mikrokontroler od renderowania fontu i znacząco upraszcza sterowanie.

Sterownik HD44780 został opracowany przez firmę Hitachi w latach 80. XX wieku i jest używany do dziś. Oprócz znaków ASCII niektóre wersje są w stanie wyświetlać również inne symbole, np.japońskie pismo sylabiczne kana. Dostępne są wyświetlacze LCD, OLED i VFD zgodne z interfejsem HD44780.

W tym przykładzie komunikacja ze sterownikiem została uproszczona do minimum. Szyna komunikacyjna została zredukowana z 8 bitów do 4 bitów, a linię $R/\overline{W}$ zwarto na stałe do masy, skutkiem czego niemożliwy jest odczyt odpowiedzi wyświetlacza na wysyłane komendy.

W systemach wbudowanych często wykorzystujemy uproszczone magistrale, nawet jeżeli spowalnia to komunikację lub ogranicza funkcjonalność. Pozwala to bowiem zastosować mikrokontroler o mniejszej liczbie wyprowadzeń, co obniża koszt i wymiary urządzenia.

[1] *American Standard Code for Information Interchange*.

Zadanie podstawowe

Celem zadania podstawowego jest uzupełnienia sterownika wyświetlacza, tak by dokonywał poprawnej konfiguracji i umożliwiał wyświetlanie łańcuchów znaków.

Wymagania funkcjonalne

1. Wyświetlacz wyświetla tekst `Hello, world!` począwszy od trzeciej kolumny drugiego wiersza.
2. Kursor jest niewidoczny.

Rysunek 1: Oczekiwany stan wyświetlacza

Modyfikacja programu

Konfiguracja wyświetlacza

Uzupełnij metodę `LcdDisplay::init()`. Inicjalizuje ona szynę danych `bus`, a następnie wysyła z jej pomocą 5 instrukcji do kontrolera HD44780, zgodnie z zalecaną procedurą inicjalizacji i konfiguracji. Ponieważ linia $R/\overline{W}$ nie jest podłączona, nie ma możliwości, by uzyskać z wyświetlacza informację o zakończeniu przerwarzania instrukcji. Maksymalny czas obsługi każdej z instrukcji jest zadeklarowany przez producenta sterownika, więc wystarczy po każdej z 5 komend `bus.instructionWrite()` odmierzyć odpowiedni interwał.

Skorzystaj z tabeli *Instructions* dokumentacji kontrolera HD44780.

Ponieważ odmierzamy krótkie odcinki czasu i to tylko w momencie aktualizowania wyświetlacza, nieopłacalne jest implementowanie ich odmierzania za pomocą timera. Użyj funkcji `_delay_us()` i `_delay_ms()`.

Przesuwanie kursora

Uzupełnij metodę `LcdDisplay::goTo(uint8_t line, uint8_t column)`, by przesunąć kursor na wyznaczoną pozycję za pomocą instrukcji *Set DDRAM address*, która ustawia adres pozycji w pamięci *Display Data*.

Niezależnie od fizycznej szerokości wyświetlacza pojedyncza linia ma długość 64 (`0x40`) znaków.

Kod instrukcji *Set DDRAM addres* został zdefiniowany w pliku `lcdDisplayInstruction.hpp` jako `SET_DDRAM_ADDRESS`.

Na tym etapie na wyświetlaczu będzie widoczna litera *a* wypisana przez polecenie w funkcji `main()`:

```
1 lcdDisplay.write('a');
```

Rysunek 2: Stan wyświetlacza

Przesuń kursor na odpowiednią pozycję, dopisując w funkcji `main()`:

```
1 lcdDisplay.goTo(1, 2);
```

Wyświetlanie łańcucha znaków

Uzupełnij metodę `LcdDisplay::write(const char* text)`, by wyświetlić łańcuch znaków w funkcji `main()` za pomocą:

```
1 lcdDisplay.write("Hello, world!");
```

Zwróć uwagę, że łańcuch znaków w językach C i C++[2] to tablica znaków zakończonych wartością `0`, zatem literał łańcuchowy zapisany za pomocą cudzysłowów:

```
1 const char* foo = "Hello!";
```

może być traktowany jako tablica znaków[3]:

```
1 const char foo[] = {'H', 'e', 'l', 'l', 'o', '!', 0};
```

[2] W języku C++ istnieje jeszcze klasa `std::string`, która ma znacznie większe możliwości przetwarzania tekstu, ale nie jest zaimplementowana w kompilatorze AVR-GCC.

[3] Zwróć uwagę, że zapisanie łańcucha znaków o długości N wymaga $N + 1$ bajtów.

Usunięcie kursora

Zmodyfikuj opcje instrukcji *Display on/off control* (`DISPLAY_CONTROL`) w metodzie `LcdDisplay::init()`, by wyłączyć wyświetlanie kursora.

Zadanie rozszerzone

Zdefiniuj dwa własne symbole i wyświetl je na początku wyświetlacza.

Wymagania funkcjonalne

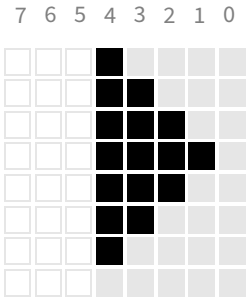
1. Oprócz napisu `Hello, world!` na początku wyświetlacza wyświetlane są zdefiniowane przez Ciebie symbole.

Rysunek 3: Przykładowy stan wyświetlacza

Modyfikacja programu

Projektowanie symboli

Zaprojektuj dwa symbole, według poniższego schematu. Skorzystaj z szablonu zamieszczonego w sprawozdaniu.



Rysunek 4: Przykład własnego symbolu

```
constexpr uint8_t RIGHT_TRIANGLE[] = {
    0b00010000,
    0b00011000,
    0b00011100,
```

```
    0b00011110,
    0b00011100,
    0b00011000,
    0b00010000,
    0b00000000,
};
```

Dodawanie symboli do sterownika

Dodaj zaprojektowane symbole do pamięci CGRAM, przekazując je do metody `LcdDisplay::addSymbol()` jako symbole o numerach `0` i `1`, np.:

```
lcdDisplay.addSymbol(RIGHT_TRIANGLE, 0);
lcdDisplay.addSymbol(LEFT_TRIANGLE, 1);
```

Uzupełnij metodę `LcdDisplay::addSymbol()`, tak by zapisywała tablicę, otrzymaną w argumencie `symbol`, w pamięci CGRAM (*Character Generator RAM*) pod adresem zależnym od argumentu `position`. Użyj instrukcji *Set CGRAM address*, by ustawić adres, i metody `bus.dataWrite()`[4], by zapisać dane.

Bajty symboli są ułożone w pamięci CGRAM jeden po drugim. Kolejne bajty pierwszego symbolu należy zatem zapisywać pod adresami `0`…`7`, a drugiego — `8`…`15`.

Użyj pętli **`for`**`()`, by powtórzyć operacje adresowania i zapisu dla każdego bajtu symbolu.

Kod instrukcji *Set CGRAM addres* został zdefiniowany w pliku `lcdDisplayInstruction.hpp` jako `SET_CGRAM_ADDRESS`.

Wyświetlanie symboli

Wyświetl zdefiniowane symbole w funkcji `main()`. Numer znaku możesz wskazać w postaci liczbowej za pomocą sekwencji `\x`:

```
lcdDisplay.write('\x00');
lcdDisplay.write('\x01');
```

[4] Tak, tej samej, której używa się do zapisywania tekstu w pamięci DDRAM.