Podstawy Systemów Mikroprocesorowych

Porty wejścia-wyjścia GPIO v.0.1

Dariusz Tefelski

2020-10-11

Spis treści

Ogólna struktura programu

Plik: **main.c** zawiera kod źródłowy, od którego zaczyna się wykonywanie programu użytkownika (funkcja main):

```
#include <avr/io.h>  //Zawsze w przypadku mikrokontrolerów AVR
#include <.....> //dołączenie plików nagłówkowych odpowiednich
↪  bibliotek......

//deklaracja i definicja zmiennych globalnych

//deklaracje z użyciem preprocesora (dyrektywa #define).........

//Procedury obsługi przerwań........

int main(void){ //funkcja główna.......
    // instrukcje   wykonywane   jednokrotnie   w   momencie   włączenia
    // układu, np.   konfiguracja układów peryferyjnych etc.
    // inicjalizacja mikrokontrolera i układów peryferyjnych........

    while(1){   // lub for(;;) lub do{ ... } while ...
        //pętla główna programu
        //instrukcje wykonywane cyklicznie lub instrukcja pusta
    }
    return 1;
}
```

Definicje własnych bibliotek wykonuje się z wykorzystaniem dyrektyw preprocesora(kompilacji warunkowej): #ifndef, #define, #endif

Laboratorium 1

Porty wejścia/wyjścia - GPIO (General Purpose Input / Output)

Ćwiczenie składa się z 4 części mających na celu głębsze zapoznanie się ze środowiskiem służącym do oprogramowania mikrokontrolerów z rodziny AVR i stworzenia programu wykorzystującego porty I/O do sterowania najprostszymi urządzeniami jak diody LED, silnik krokowy oraz odebrania informacji z przycisków z wykorzystaniem portów GPIO.

Wykonanie ćwiczenia:

1. Zaświecenie diody LED z wykorzystaniem portu mikrokontrolera.
    a. Odszukać w dokumentacji układu rejestry odpowiedzialne za konfigurację portów.
    b. Odszukać informację w nocie mikrokontrolera AVR **ATmega32** o sposobie zapisywania i odczytywania rejestrów mikrokontrolera.
    c. Zapoznać się z dokumentacją biblioteki **avr-libc**.
    d. Odszukać podłączenia diod LED w instrukcji do płytki prototypowej **Instruckcja-EvB5.1-v1.pdf**.
    e. Podłączyć kolejne wyprowadzenia (katody) diod LED do wyprowadzeń portu C mikrokontrolera: od **PC0** do **PC7**.
    f. Napisać program konfigurujący port C jako wyjściowy oraz wyświetlić odpowiednią sekwencję na diodach LED wskazaną przez prowadzącego.
2. Odebranie informacji z przycisku i sterowanie diodami LED.
    a. Podłączyć dowolny switch (np. **S1**) do pierwszego wyprowadzenia portu D (**PD0**).
    b. Diody pozostają podłączone do portu C jak w pkt1.
    c. Skonfigurować port C jako wyjściowy, a port D jako wejściowy.
    d. W pętli głównej programu sprawdzać cyklicznie stan przycisku i w momencie n aciśnięcia zapalać diodę LED.
3. Miganie diody LED.
    a. Wykorzystując funkcje biblioteczne zawarte w <utils/delay.h> napisać własną bibliotekę (np.: **longdelay.h** i **longdelay.c**) zawierającą funkcję (prototyp np.: **void longdelay(uint16_t))** do realizacji długich opóźnień (powyżej 50ms).
    b. Wykorzystać ww. funkcję do zapalania i gaszenia diody LED co ok. 1s. Działanie ma być realizowane w pętli głównej.
4. Sterowanie pracą silnika krokowego.
    a. Podłączyć silnik krokowy do wyprowadzeń portu C mikrokontrolera wg zaleceń prowadzącego.
    b. Wykorzystując funkcję z punktu 3 stworzyć program, który po naciśnięciu przycisku spowoduje zadziałanie silnika krokowego.
        - Zalecane opóźnienie pomiędzy kolejnymi sekwencjami sygnałów sterujących silnikiem: 4-250ms;
        - Wykonać bistabilne sterowanie przyciskiem włączanie albo wyłączanie silnika;
        - Sterowanie kierunkiem obrotów silnika (przycisk bistabliny) - obroty w lewo i w prawo;
        - Regulacja prędkości obrotowej (2 przyciski - szybciej i wolniej w zakresie 4-250 ms.