PSM 2012

Laboratorium prowadzone jest w oparciu o mikrokontroler z rodziny ATmega16. Dokumentacja do układów stosowanych na zajęciach znajduje się w katalogu C:\Użytkownicy\Publiczny\PSM.

Na oprogramowanie składa się:

- Pakiet WinAVR (w wersji 20100110) zawierający: kompilator avr-gcc wraz ze środowiskiem umożliwiającym jego odpowiednią pracę, biblioteki avr-libc wraz z dokumentacją: C*:\WinAVR-20100110\doc\avr-libc\avr-libc-user-manual\index.html*
- Edytor „*Programmers Notepad*", wspomagający tworzenie programów i proces kompilacji z użyciem avr-gcc oraz współpracujące z programatorem sprzętowym AVTProg1.

Ogólna struktura programu:

```
#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 – I/O.

Ćwiczenie składa się z 4 części mających na celu 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 ww. portów.

Wykonanie ćwiczenia:

1 Zapalenie diody z wykorzystaniem portu mikrokontrolera.
   a. Odszukać w dokumentacji układu rejestry odpowiedzialne za konfigurację portów.
   b. Odszukać informację w dokumentacji biblioteki avr/io.h o sposobie zapisywania i odczytywania rejestrów mikrokontrolera.
   c. Podłączyć wyprowadzenia (katody) diod LED do wyprowadzeń portu C mikrokontrolera: D0-D7 -> PC0 – PC7

d. 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. SW0) do pierwszego wyprowadzenia portu D.
   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 naciśnięcia 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 (lub gotowe funkcje biblioteczne do realizacji opóźnień) stworzyć program, który po naciśnięciu przycisku spowoduje zadziałanie silnika krokowego.
      i. Zalecane opóźnienie pomiędzy kolejnymi sekwencjami sygnałów sterujących silnikiem: 2-255ms
   c. W miarę możliwości wykonać sterowanie silnikiem w stronę przeciwną (po naciśnięciu drugiego przycisku), regulację prędkości obrotowej, oprogramować przycisk aby pracował jak przycisk bistabilny (włącz-wyłącz).