Raport do zadania 2

Imię i nazwisko: Jakub Dmochowski

Numer Albumu: 169236

Konfiguracja sprzętowa

```
#pragma config POSCMOD = XT
#pragma config OSCIOFNC = ON
#pragma config FCKSM = CSDCMD
#pragma config FNOSC = PRI
#pragma config IESO = ON
```

POSCMOD = XT - wybór zewnętrznego kwarcu jako źródła zegara

OSCIOFNC = ON - funkcja wyjścia oscylatora włączona

FCKSM = CSDCMD - wyłączenie monitorowania zegara i przełączania

FNOSC = PRI - Podstawowe źródło zegara jako domyślne

IESO = ON - włączenie wewnętrznego/zewnętrznego przełączania oscylatora

Watchdog timer, debugowanie

```
#pragma config WDTPS = PS32768
#pragma config FWPSA = PR128
#pragma config WINDIS = ON
#pragma config FWDTEN = OFF
#pragma config ICS = PGx2
#pragma config GWRP = OFF
#pragma config GCP = OFF
#pragma config JTAGEN = OFF
```

WDTPS = PS32768 - Prescaler watchdog timera ustawiony na 1:32768

FWPSA = PR128 - Prescaler A watchdog timera ustawiony na 1:128

WINDIS = ON - Windowed watchdog timer wyłączony

FWDTEN = OFF - Watchdog timer całkowicie wyłączony

ICS = PGx2 - Komunikacja z debuggerem przez piny PGC2/PGD2

GWRP = OFF - Ochrona zapisu do pamięci programu wyłączona

GCP = OFF - Ochrona odczytu kodu wyłączona

JTAGEN = OFF - Interfejs JTAG wyłączony

Biblioteki

```
#include <xc.h>
#include <libpic30.h>
#include <stdbool.h>
```

#include <xc.h> - Główna biblioteka dla kompilatorów XC

#include <libpic30.h> - Biblioteka specyficzna dla PIC30, zawiera funkcje opóźnień

#include <stdbool.h> - Biblioteka standardowa C dla typu bool (true/false

Definicje i zmienne globalne

```
#define FCY 4000000UL
#define DEBOUNCE_DELAY 200000
#define buttonnext PORTDbits.RD6
#define buttonprev PORTDbits.RD13

unsigned char tryb = 1;
unsigned long delay_value = 500000;
```

#define FCY 4000000UL - Częstotliwość procesora 4 MHz

#define DEBOUNCE_DELAY 200000 - Opóźnienie dla eliminacji drgań styków

#define buttonnext PORTDbits.RD6 - Przycisk "następny"

#define buttonprev PORTDbits.RD13 - Przycisk "poprzedni"

unsigned char tryb = 1; - Tryb pracy (1 lub 2)

unsigned long delay_value = 500000; - Zmienna wartość opóźnienia sterowana potencjometrem

Konfiguracja wejść/wyjść i ADC

```
TRISA = 0x0000;
TRISD = 0xFFFF;

AD1PCFG = 0xFFDF;
AD1CON1 = 0x00E0;
AD1CON2 = 0;
AD1CON3 = 0x1F3F;
AD1CHS = 5;
AD1CON1bits.ADON = 1;
```

TRISA = 0x0000; - Port A jako wyjście (sterowanie LED)

TRISD = 0xFFFF; - Port D jako wejście (odczyt przycisków)

AD1PCFG = 0xFFDF; - Pin AN5 jako analogowy, pozostałe cyfrowe (bit 5 = 0)

AD1CON1 = 0x00E0; - Format danych, tryb próbkowania

AD1CON2 = 0; - Referencja napięcia i tryb skanowania

AD1CON3 = 0x1F3F; - Czas konwersji i próbkowania

AD1CHS = 5 - Wybór kanału AN5 dla konwersji

AD1CON1bits.ADON = 1 - Włączenie przetwornika ADC

potencjometr() - Odczyt wartości z przetwornika ADC

```
uint16_t potencjometr() {
    AD1CON1bits.SAMP = 1;
    __delay32(100);
    AD1CON1bits.SAMP = 0;
    while (!AD1CON1bits.DONE);
    return ADC1BUF0;
}
```

Definicje:

- AD1CON1bits.SAMP = 1; - Rozpoczęcie próbkowania
- __delay32(100); - Czas stabilizacji próbkowania
- AD1CON1bits.SAMP = 0; - Zakończenie próbkowania, start konwersji
- while (!AD1CON1bits.DONE); - Oczekiwanie na zakończenie konwersji
- return ADC1BUF0; - Zwrócenie wyniku konwersji (0-1023)

Zasada działania:

- Rozpoczyna próbkowanie sygnału analogowego z potencjometru
- Czeka 100 cykli na stabilizację
- Kończy próbkowanie i rozpoczyna konwersję ADC
- Oczekuje na zakończenie konwersji
- Zwraca 10-bitową wartość (0-1023) z bufora

predkosc() - Kontrola prędkości na podstawie potencjometru

```
void predkosc() {
    uint16_t pot_value = potencjometr();

    if (pot_value < 205) {
        delay_value = 1000000;
    } else if (pot_value < 410) {
        delay_value = 750000;
    } else if (pot_value < 615) {
        delay_value = 500000;
    } else if (pot_value < 820) {
        delay_value = 250000;
    } else {
        delay_value = 100000;
    }
}
```

Zasada działania

- Odczytuje wartość z potencjometru (0-1023)
- Dzieli zakres na 5 przedziałów o równej szerokości (~20% każdy)
- Ustawia wartość opóźnienia odwrotnie proporcjonalnie do pozycji potencjometru
- Im wyższa wartość potencjometru, tym mniejsze opóźnienie (wyższa prędkość)

funk3() - 8-bitowy licznik Gray z kontrolą prędkości

```
void funk3() {
    unsigned char licznik_normalny = 0;
    unsigned char wynik_gray;

    while (1) {
        predkosc();

        wynik_gray = licznik_normalny ^ (licznik_normalny >> 1);

        LATA = wynik_gray;
        __delay32(delay_value);

        if (!buttonnext || !buttonprev) {
            return;
        }
        licznik_normalny = licznik_normalny + 1;
    }
}
```

Definicje:

- predkosc(); - Aktualizacja prędkości na podstawie potencjometru
- __delay32(delay_value); - Opóźnienie sterowane potencjometreu

Zasada działania:

- Implementuje 8-bitowy licznik Gray zliczający w górę
- W każdej iteracji wywołuje predkosc() dla aktualizacji tempa
- Konwertuje liczbę binarną do kodu Gray: wynik_gray = licznik_normalny ^ (licznik_normalny >> 1)
- Opóźnienie zmienne w zakresie 100000-1000000 cykli w zależności od potencjometru
- Kończy działanie po wciśnięciu dowolnego przycisku

funk7() - Wężyk z kontrolą prędkości

```
void funk7() {
    unsigned char wezyk;
    unsigned char kierunek_w_prawo = 1;

    wezyk = 0b00000111;

    while (1) {
        predkosc();

        LATA = wezyk;
        __delay32(delay_value);

        if (!buttonnext || !buttonprev) {
            return;
        }

        if (kierunek_w_prawo == 1) {
            wezyk = wezyk << 1;

            if (wezyk == 0b11100000) {
                kierunek_w_prawo = 0;
            }
        } else {
            wezyk = wezyk >> 1;

            if (wezyk == 0b00000111) {
                kierunek_w_prawo = 1;
            }
        }
    }
}
```

Definicje:

- predkosc(); - Aktualizacja prędkości na podstawie potencjometru
- __delay32(delay_value); - Opóźnienie sterowane potencjometrem
- wezyk = wezyk << 1; - Przesunięcie w lewo
- kierunek_w_prawo = 0; - Zmiana kierunku
- wezyk = wezyk >> 1- Przesunięcie w prawo
- kierunek_w_prawo = 1; - Zmiana kierunku

Zasada działania:

- Implementuje funkcje wężyk
- W każdej iteracji wywołuje predkosc() dla aktualizacji tempa
- Rozpoczyna z wzorem 0b00000111 (3 LED po prawej stronie)
- Przesuwa wzór w lewo do momentu osiągnięcia 0b11100000
- Następnie zmienia kierunek i przesuwa w prawo do 0b00000111
- Oscyluje między skrajnymi pozycjami
- Opóźnienie zmienne sterowane potencjometrem

Pętla while w main()

```
while (1) {
    switch (tryb) {
        case 1:
            funk3();
            break;
        case 2:
            funk7();
            break;
    }
```

Program działa w nieskończonej pętli, wykonując wybraną funkcję do momentu wciśnięcia przycisku, sprawdza który przycisk został wciśnięty po czym zmienia tryb.

- 1 - Licznik Gray
- 2 - Wężyk

Debouncing

```
        if (!buttonnext) {
            __delay32(DEBOUNCE_DELAY);
            if (!buttonnext) {
                tryb++;
                if (tryb > 2)
                    tryb = 1;
                while (!buttonnext);
                __delay32(DEBOUNCE_DELAY);
            }
        }

        if (!buttonprev) {
            __delay32(DEBOUNCE_DELAY);
            if (!buttonprev) {
                if (tryb == 1)
                    tryb = 2;
                else
                    tryb--;
                while (!buttonprev);
                __delay32(DEBOUNCE_DELAY);
            }
        }
    }
}
```

RD6:

- Przełącza z trybu 1 na 2, z trybu 2 na 1
- Implementuje debouncing z opóźnieniem 200000 cykli

RD13:

- Przełącza z trybu 2 na 1, z trybu 1 na 2
- Implementuje debouncing z opóźnieniem 200000 cykli