169224_zad_4 – raport z zadania 4

Jakub Budzich – nr albumu 169224

Platforma: MPLAB X IDE + XC16
Mikrokontroler: PIC24FJ128GA010

Opis funkcjonalności programów

Z racji że w zadaniu 4 mieliśmy wybór co do wykonania zadania -- wybrałem wariant z **kuchenką mikrofalową**. Program symuluje działanie wyświetlacza kuchenki mikrofalowej. Program umożliwia ustawienie czasu (+10 sekund lub +1 minuta), uruchamianie odliczania w dół nastawionego czasu oraz zatrzymywanie czasu. Po zakończeniu odliczania system automatycznie wraca do stanu początkowego po 5 sekundach. Informacje wyświetlane są na wyświetlaczu LCD dzięki plikom lcd.h i lcd.c

Stan	Funkcja	Opis działania
0	Stan spoczynku	Kuchenka zatrzymana, możliwość ustawiania czasu – GOTOWE ZA: ; CZAS: 00:00
1	Stan pracy	Aktywne odliczanie czasu w dół, Pracuje ; Czas odliczany w dół zgodnie z nastawą
2	Stan pauza	Odliczanie zatrzymane, możliwość wznowienia ponownym kliknięciem Pauza ; Czas: *„pozostały czas z nastawy”*

Obsługa przycisków

Obsługa przycisków odbywa się przez przerwanie _CNInterrupt() (Change Notification), 3 przyciski są wykorzystywane do obsługi kuchenki:

- **RD6** → Dodanie 1 minuty (60 sekund) do czasu
- **RD7** → Dodanie 10 sekund do czasu
- **RD13** → Start odliczania/ Pauza odliczania (kuchenka zatrzymuje zegar i czeka na wznowienie tym samym przyciskiem)

Wyjaśnienie poszczególnych lini kodu

0. Konfiguracja sprzętowa i inicjalizacja

- `POSCMOD = NONE` - wybór trybu oscylatora (brak zewnętrznego oscylatora)
- `OSCIOFNC = ON` - funkcja wyjścia oscylatora włączona
- `FCKSM = CSDCMD` - przełączanie zegara i monitor wyłączone
- `FNOSC = FRC` - wybór wewnętrznego oscylatora RC
- `IESO = OFF` - tryb przełączania między oscylatorami wyłączony
- `WDTPS = PS32768`, `FWPSA = PR128`, `WINDIS = ON`, `FWDTEN = OFF` - konfiguracja watchdog timera (wyłączony)
- `ICS = PGx2` - wybór kanału komunikacyjnego dla debuggera
- `GWRP = OFF`, `GCP = OFF` - wyłączenie ochrony pamięci programu
- `JTAGEN = OFF` - wyłączenie portu JTAG

```
#include <stdio.h>       // do printf i sprintf
#include <stdlib.h>      // podstawowe funkcje
#include <xc.h>          // funkcje mikrokontrolera
#include <libpic30.h>    // do opoznien
#include "p24FJ128GA010.h"  // definicje pinow
#include "lcd.h"         // obsluga wyswietlacza z pliku
```

- **stdio.h**: biblioteka do funkcji printf i sprintf – do obsługi stringów na ekranie
- **stdlib.h**: podstawowe funkcje w c
- **xc.h**: plik z funkcjami mikrokontrolera na którym wykonywany jest program, zawiera on defincje mikrokontrolera dla kompilatora XC16
- **libpic30.h**: biblioteka do delay w której jest funkcja
- **p24FJ128GA010.h:** definicje pinów dla mikrokontrolera PIC24FJ128GA010
- **lcd.h:** plik nagłówkowy do obsługi wyświetlacza

```
// deklaracja zegara systemowego
#define XTAL_FREQ 8000000
#define FCY 4000000

// Komendy dla LCD
#define LCD_CLEAR 0x01
#define LCD_HOME 0x02
#define LCD_CURSOR_OFF 0x0C

// Przyciski - ktory co robi
#define PRZYCISK_1MIN PORTDbits.RD6     // dodaj 1 minute
#define PRZYCISK_10SEC PORTDbits.RD7    // dodaj 10 sekund
#define PRZYCISK_START PORTAbits.RD13    // start/stop
```

```
void sprawdz_czas(void);
void ustaw_urzadzenie(void);
void pokaz_na_ekranie(void);
void zatrzymaj(void);
void zacznij(void);
```

- **XTAL_FREQ:** Definicja częstotliwości oscylatora - 8MHz (używana przez libpic30)
- **FCY:** Częstotliwość cyklu instrukcji - FCY = XTAL_FREQ/2 = 4MHz
- **LCD_CLEAR:** Komenda dla wyświetlacza LCD – czyszczenie wyświetlacza i ustawianie kursora na pozycję (0,0)
- **LCD_HOME:** Kolejna komenda dla LCD – ustawia kursor na pozycję (0,0)
- **LCD_CURSOR_OFF:** Znów dla LCD komenda – wyłącza widoczność kursora
- **PRZYCISK_1MIN:** odczyt stanu przycisku z portu RD6 – dodaje 1 minutę do czasu
- **PRZYCISK_10SEC:** odczyt stanu przycisku z portu RD7 – dodaje 10 sekund do czasu
- **PRZYCISK_START:** odczyt stanu przycisku z portu RD13 – start odliczania kuchenki / pauza w odliczaniu kuchenki
- **PRZYCISK_RESET:** odczyt stanu przycisku z portu RA7 – reset kuchenki, powrót do stanu 0 początkowego
- Dodano też deklaracje funkcji

Wyjaśnienie działania Przycisku 1MIN i 10 SEKUND – można dodawać czas jak jest stan kuchneki stan pracy – postanowiłem to tak rozwiązać bo w analogowych kuchenkach można było dodawać czas podczas pracy zwiększając wartość pokrętła

1. Zmienne globalne

```
volatile uint16_t czas_sekundy = 0;           // ile sekund zostało
volatile uint8_t stan = 0;                    // 0=stop, 1=dziala, 2=pauza
volatile uint16_t odswiez_ekran = 1;          // czy odswiezyc wyświetlacz
volatile uint16_t migaj = 0;                  // do migania dwukropka
volatile uint16_t licznik_ms = 0;             // licznik milisekund
volatile uint16_t ostatnia_sekunda = 0;       // kiedy ostatnio odliczylismy
sekundę
volatile uint16_t skonczyl = 0;               // czy skonczylo sie odliczanie
```

- **czas_sekundy:** zmienna przechowuje pozostały czas ile zostało do odliczenia w dół
- **stan:** zmienna stanu kuchenki: 0= zatrzymana (początkowy stan), 1 = praca (odliczanie w dół), 2 = pauza (stop czasu odliczania)

- **odswiez_ekran:** flaga mówiąca programowi że trzeba odświeżyć ekran jeśli stan jest 1
- **migaj:** zmienna dla efektu migającego dwukropka (przyjmuje wartość 0 lub 1)
- **licznik_ms:** licznik miliseknud używany w przerwaniu Timer1
- **ostatnia_sekunda:** znacznik czasu ostatniego odliczenia sekundy
- **skonczyl:** flaga informująca o zakończeniu odliczania – 0 = nie, 1 = tak

Opóźnienie realizowane jest przez zagnieżdżone pętle i instrukcję asm("NOP")

2. Funkcja przerwania `_T1Interrupt()`

```
void __attribute__((interrupt, auto_psv)) _T1Interrupt(void)
{
    IFS0bits.T1IF = 0;          // wyczysc flage przerwania

    licznik_ms++;               // zwieksz licznik milisekund

    // Co 500ms zmien miganie
    if (licznik_ms % 500 == 0) {
        migaj = !migaj;
    }
}
```

- Definicja procedury przerwania Timer1
- **auto_psv** - automatyczne zapisywanie rejestrów
- **IFS0bits.T1IF = 0;** - wyzerowanie flagi przerwania Timer1 (obowiązkowe w każdym przerwaniu)
- licznik_ms++ - inkrementacja licznika milisekund (wywoływane co 1ms)
- Co 500 ms zmieniany jest stan migaj dla migającego dwukropka

3. Funkcja przerwania `_CNInterrupt()` - obsługa przycisków

```
void __attribute__((interrupt, no_auto_psv)) _CNInterrupt(void) {
    __delay32(FCY/100);  //debouncing 10ms

    // Przycisk +1min (RD6/CN15)
    if(PORTDbits.RD6 == 0) {
        czas_sekundy += 60;     // dodaj 60 sekund = 1 min
        if (czas_sekundy > 5999) czas_sekundy = 5999; // max 99:59
        odswiez_ekran = 1;      // odswiez ekran
    }

    // Przycisk +10sec (RD7/CN16)
    else if(PORTDbits.RD7 == 0) {
        czas_sekundy += 10;     // dodaj 10 sekund
```

```
        if (czas_sekundy > 5999) czas_sekundy = 5999; // max 99:59
        odswiez_ekran = 1;      // odswiez ekran
    }

    // Przycisk Start/Stop (RD13/CN19)
    else if(PORTDbits.RD13 == 0) {
        if (stan == 0 || stan == 2) {      // jesli zatrzymana lub pauza
            if (czas_sekundy > 0) {         // jesli jest czas do odliczenia
                zacznij();                  // zacznij odliczanie
            }
        } else if (stan == 1) {             // jesli dziala
            stan = 2;                       // ustaw na pauze
            odswiez_ekran = 1;             // odswiez ekran
        }
    }

    // Czekaj na zwolnienie przycisk□w
    while(PORTDbits.RD6 == 0 || PORTDbits.RD7 == 0 || PORTDbits.RD13 == 0);

    // Wyczysc flage przerwania
    IFS1bits.CNIF = 0;
}
```

- **Change Notification interrupt** - przerwanie wywoływane przy zmianie stanu pinów
- **__delay32(FCY/100)** - debouncing, eliminacja drgań styków około 10ms opóźnienia
- **no_auto_psv** - brak automatycznego zapisywania rejestrów (krótkie przerwanie)
- Obsługa wszystkich przycisków w jednej funkcji przerwania
- **while()** - oczekiwanie na zwolnienie przycisków
- **IFS1bits.CNIF = 0** - wyzerowanie flagi przerwania CN

4. Funkcja `main( )`

```
int main(void)
{
    ustaw_urzadzenie();

    while (1) {
        sprawdz_czas();         // sprawdz czy minela sekunda

        if (odswiez_ekran) {    // jesli trzeba odswiezyc
            pokaz_na_ekranie();  // pokaz aktualny stan
            odswiez_ekran = 0;
        }

        // Automatyczne resetowanie po 5 sekundach od zakonczenia
```

```
        static uint16_t czas_gotowe = 0;
        if (czas_sekundy == 0 && stan == 0 && skonczyl) {
            czas_gotowe++;
            if (czas_gotowe > 5000) {  // po 5 sekundach (5000ms)
                skonczyl = 0;
                czas_gotowe = 0;
                odswiez_ekran = 1;
            }
        } else {
            czas_gotowe = 0;
        }
    }

    return 0;
}
```

- Wywoływana jest inicjalizacja urządzenia
- W pętli która jest nieskończona
  - Sprawdzany jest stan przycisków
  - Sprawdzane jest czy minęła sekunda odliczania
  - Odświeżany jest wyświetialcz jeśli jest to potrzebne (np. jak zmiana czasu nastąpi poprzez odliczanie, zmiana stanu, interakcja przyciskiem coś zmieni)
  - Automatyczne resetowanie po zakończeniu gotowania (po 5 sekundach)

5. Funkcja `sprawdz_czas( )`

```
// Sprawdza czy minela sekunda i odlicza czas
void sprawdz_czas(void)
{
    // Jesli kuchenka dziala i jest czas do odliczenia
    if (stan == 1 && czas_sekundy > 0) {
        // Jesli minelo 1000ms (1 sekunda)
        if (licznik_ms - ostatnia_sekunda >= 1000) {
            czas_sekundy--;             // odlicz sekunde
            ostatnia_sekunda = licznik_ms; // zapamietaj kiedy
            odswiez_ekran = 1;         // odswiez ekran

            // Jesli czas sie skonczyl
            if (czas_sekundy == 0) {
                zatrzymaj();            // zatrzymaj kuchenke bo juz koniec
            }
        }
    }
}
```

- Sprawdza czy kuchenka działa (stan == 1) i czy jest czas do odliczenia

- Co 1 sekunde (1000 ms) zmienia czas odliczania i odświeża wyświetlacz LCD
- Jeśli czas osiągnie 0 to timer jest zatrzymywany
- Używana jest róznica z ***licznik_ms – ostatnia_sekunda*** dla precyzyjnego pomiaru

6. Funkcja `ustaw_urzadzenie( )`

```
void ustaw_urzadzenie(void)
{
    AD1PCFG = 0xFFFF;           // wszystkie piny cyfrowe
    TRISA = 0x0000;             // Port A jako wyjscie (dla LCD)
    TRISD = 0xFFFF;             // Port D jako wejscie (przyciski)

    // Konfiguracja Change Notification interrupts
    CNPU1bits.CN15PUE = 1;      // Pull-up dla RD6 (+1min)
    CNPU2bits.CN16PUE = 1;      // Pull-up dla RD7 (+10sec)
    CNPU2bits.CN19PUE = 1;      // Pull-up dla RD13 (Start/Stop)

    // Wlaczenie przerwan Change Notification
    CNEN1bits.CN15IE = 1;       // Wlacz przerwanie dla RD6
    CNEN2bits.CN16IE = 1;       // Wlacz przerwanie dla RD7
    CNEN2bits.CN19IE = 1;       // Wlacz przerwanie dla RD13

    IFS1bits.CNIF = 0;          // Wyczysc flage przerwania CN
    IEC1bits.CNIE = 1;          // Wlacz przerwania CN

    // Uruchomienie LCD
    LCD_Initialize();
    LCD_ClearScreen();

    // Ustaw timer na 1ms
    T1CON = 0;                  // wyczysc ustawienia timera
    TMR1 = 0;                   // wyczysc licznik
    PR1 = FCY/1000 - 1;         // POPRAWIONE - ustaw na 1ms
    T1CONbits.TCKPS = 0b00;     // bez dzielnika
    IPC0bits.T1IP = 3;          // priorytet 3 (nizszy niz CN)
    IFS0bits.T1IF = 0;          // wyczysc flage
    IEC0bits.T1IE = 1;          // wlacz przerwanie
    T1CONbits.TON = 1;          // wlacz timer

    // Tekst poczatkowy
    LCD_PutString("GOTOWE ZA:", 10);
    LCD_PutChar('\n');
    LCD_PutString("Czas: 00:00", 11);
    // Wlacz przerwania globalne
    INTCON1bits.NSTDIS = 0;
}
```

- **AD1PCFG = 0xFFFF** - wszystkie piny jako cyfrowe
- **CNPU1bits/CNPU2bits** - włączenie wewnętrznych rezystorów pull-up
- **CNEN1bits/CNEN2bits** - włączenie przerwań CN dla konkretnych pinów
- **IEC1bits.CNIE = 1** - włączenie przerwań Change Notification globalnie

7. Funkcja `pokaz_na_ekranie( )`

```
void pokaz_na_ekranie(void)
{
    char tekst[17];
    uint16_t minuty = czas_sekundy / 60;     // ile minut
    uint16_t sekundy = czas_sekundy % 60;    // ile sekund

    LCD_ClearScreen();                  // wyczysc ekran

    if (stan == 0) {                    // stan - zatrzymana kuchenka
        if (czas_sekundy == 0) {
            if (skonczyl) {             // jesli skonczylo sie odliczanie
                LCD_PutString("GOTOWE", 6); // wyswietlamy napis GOTOWE
            } else {                    // powrot do poczatkowego stanu
                LCD_PutString("GOTOWE ZA:", 10);
            }
        } else {
            LCD_PutString("GOTOWE ZA:", 10);
        }
    } else if (stan == 1) {             // stan dzialania - po wybraniu czasu
juz
        LCD_PutString("Pracuje", 7);
        skonczyl = 1;
    } else if (stan == 2) {             // stan pauza
        LCD_PutString("Pauza", 5);
    }

    LCD_PutChar('\n');

    // drugi wiersz na wyswietlaczu lcd
    // wyswietlany jest czas badz druga czesc komunikatu koncowego
    if (stan == 0 && czas_sekundy == 0 && skonczyl) {
        LCD_PutString("SMACZNEGO!", 10); // komunikat konca
    } else {
        if (stan == 2 && migaj) {
            sprintf(tekst, "Czas: %02d %02d", minuty, sekundy);
        } else {
            sprintf(tekst, "Czas: %02d:%02d", minuty, sekundy);
        }
```

```
        LCD_PutString(tekst, 11);
    }
}
```

- Funkcja odpowiada za wyświetlanie aktualnego stanu na ekranie LCD
- Pierwszy wiersz LCD:
  - Stan 0: „GOTOWE ZA” lub „GOTOWE” – po zakończeniu odliczania
  - Stan 1: „Pracuje”
  - Stan 2: „Pauza”
- Drugi wiersz LCD:
  - Stan 0 i 1: „Czas Minuty:Sekundy MM:SS” – aktualny czas pracy pozostały
  - Stan po zakończeniu: „SMACZNEGO” – dalsza część komunikatu końcowego

8. Funkcje sterujące – `zatrzymaj( ), zacznij( )`

```
void zatrzymaj(void)
{
    stan = 0;                    // ustaw stan na zatrzymana
    odswiez_ekran = 1;          // odswiez ekran
}

void zacznij(void)
{
    stan = 1;                            // stan - dziala
    ostatnia_sekunda = licznik_ms;       // zapamietaj czas startu
    odswiez_ekran = 1;                  // odswiez ekran
}
```

- **zatrzymaj():** przełącza timer w stan spoczynku (zatrzymany czas), odswiezany ekran
- **zacznij():** rozpoczyna odliczanie i zapisuje czas startu poprzez odjęcie **ostatnia_sekunda = licznik_ms;**

10. Cały kod programu:

```
/*
 * File:   main.c
 * Author: Jakub Budzich - ISI1
 *
 * Created on May 19, 2025, 9:14
 */
```

```
#pragma config POSCMOD = NONE
#pragma config OSCIOFNC = ON
#pragma config FCKSM = CSDCMD
#pragma config FNOSC = FRC
#pragma config IESO = OFF
#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

#include <stdio.h>
#include <stdlib.h>
#include <xc.h>
#include <libpic30.h>
#include "lcd.h"

// Deklaracja zegara systemowego
#define XTAL_FREQ 8000000
#define FCY 4000000

// DEKLARACJE FUNKCJI
void sprawdz_czas(void);
void ustaw_urzadzenie(void);
void pokaz_na_ekranie(void);
void zatrzymaj(void);
void zacznij(void);

// Zmienne globalne
volatile uint16_t czas_sekundy = 0;           // ile sekund zostalo
volatile uint8_t stan = 0;                    // 0=stop, 1=dziala, 2=pauza
volatile uint16_t odswiez_ekran = 1;          // czy odswiezyc wyswietlacz
volatile uint16_t migaj = 0;                  // do migania dwukropka
volatile uint16_t licznik_ms = 0;             // licznik milisekund
volatile uint16_t ostatnia_sekunda = 0;       // kiedy ostatnio odliczylismy
sekunde
volatile uint16_t skonczyl = 0;               // czy skonczylo sie odliczanie

// Przerwanie od Timer1 (1ms)
void __attribute__((interrupt, auto_psv)) _T1Interrupt(void)
{
    IFS0bits.T1IF = 0;          // wyczysc flage przerwania

    licznik_ms++;               // zwieksz licznik milisekund
```

```
    // Co 500ms zmien miganie
    if (licznik_ms % 500 == 0) {
        migaj = !migaj;
    }
}

// Przerwanie Change Notification - obs?uga przyciskow
void __attribute__((interrupt, no_auto_psv)) _CNInterrupt(void) {
    __delay32(FCY/100);  //debouncing 10ms

    // Przycisk +1min (RD6/CN15)
    if(PORTDbits.RD6 == 0) {
        czas_sekundy += 60;     // dodaj 60 sekund = 1 min
        if (czas_sekundy > 5999) czas_sekundy = 5999; // max 99:59
        odswiez_ekran = 1;      // odswiez ekran
    }

    // Przycisk +10sec (RD7/CN16)
    else if(PORTDbits.RD7 == 0) {
        czas_sekundy += 10;     // dodaj 10 sekund
        if (czas_sekundy > 5999) czas_sekundy = 5999; // max 99:59
        odswiez_ekran = 1;      // odswiez ekran
    }

    // Przycisk Start/Stop (RD13/CN19)
    else if(PORTDbits.RD13 == 0) {
        if (stan == 0 || stan == 2) {      // jesli zatrzymana lub pauza
            if (czas_sekundy > 0) {         // jesli jest czas do odliczenia
                zacznij();                  // zacznij odliczanie
            }
        } else if (stan == 1) {             // jesli dziala
            stan = 2;                       // ustaw na pauze
            odswiez_ekran = 1;             // odswiez ekran
        }
    }

    // Czekaj na zwolnienie przycisków
    while(PORTDbits.RD6 == 0 || PORTDbits.RD7 == 0 || PORTDbits.RD13 == 0);

    // Wyczysc flage przerwania
    IFS1bits.CNIF = 0;
}

int main(void)
{
    ustaw_urzadzenie();
```

```
    while (1) {
        sprawdz_czas();         // sprawdz czy minela sekunda

        if (odswiez_ekran) {    // jesli trzeba odswiezyc
            pokaz_na_ekranie();  // pokaz aktualny stan
            odswiez_ekran = 0;
        }

        // Automatyczne resetowanie po 5 sekundach od zako?czenia
        static uint16_t czas_gotowe = 0;
        if (czas_sekundy == 0 && stan == 0 && skonczyl) {
            czas_gotowe++;
            if (czas_gotowe > 5000) {  // po 5 sekundach (5000ms)
                skonczyl = 0;
                czas_gotowe = 0;
                odswiez_ekran = 1;
            }
        } else {
            czas_gotowe = 0;
        }
    }

    return 0;
}

// Sprawdza czy minela sekunda i odlicza czas
void sprawdz_czas(void)
{
    // Jesli kuchenka dziala i jest czas do odliczenia
    if (stan == 1 && czas_sekundy > 0) {
        // Jesli minelo 1000ms (1 sekunda)
        if (licznik_ms - ostatnia_sekunda >= 1000) {
            czas_sekundy--;                 // odlicz sekunde
            ostatnia_sekunda = licznik_ms;  // zapamietaj kiedy
            odswiez_ekran = 1;             // odswiez ekran

            // Jesli czas sie skonczyl
            if (czas_sekundy == 0) {
                zatrzymaj();                // zatrzymaj kuchenke
            }
        }
    }
}

// Inicjalizacja urzadzenia
void ustaw_urzadzenie(void)
{
    AD1PCFG = 0xFFFF;           // wszystkie piny cyfrowe
```

```
    TRISA = 0x0000;             // Port A jako wyjscie (dla LCD)
    TRISD = 0xFFFF;             // Port D jako wejscie (przyciski)

    // Konfiguracja Change Notification interrupts

    CNPU1bits.CN15PUE = 1;      // Pull-up dla RD6 (+1min)
    CNPU2bits.CN16PUE = 1;      // Pull-up dla RD7 (+10sec)
    CNPU2bits.CN19PUE = 1;      // Pull-up dla RD13 (Start/Stop)

    // W??czenie przerwa? Change Notification
    CNEN1bits.CN15IE = 1;       // Wlacz przerwanie dla RD6
    CNEN2bits.CN16IE = 1;       // Wlacz przerwanie dla RD7
    CNEN2bits.CN19IE = 1;       // Wlacz przerwanie dla RD13

    IFS1bits.CNIF = 0;          // Wyczysc flage przerwania CN
    IEC1bits.CNIE = 1;          // Wlacz przerwania CN

    // Uruchomienie LCD
    LCD_Initialize();
    LCD_ClearScreen();

    // Ustaw timer na 1ms
    T1CON = 0;                  // wyczysc ustawienia timera
    TMR1 = 0;                   // wyczysc licznik
    PR1 = FCY/1000 - 1;         // POPRAWIONE - ustaw na 1ms
    T1CONbits.TCKPS = 0b00;     // bez dzielnika
    IPC0bits.T1IP = 3;          // priorytet 3 (nizszy niz CN)
    IFS0bits.T1IF = 0;          // wyczysc flage
    IEC0bits.T1IE = 1;          // wlacz przerwanie
    T1CONbits.TON = 1;          // wlacz timer

    // Tekst poczatkowy
    LCD_PutString("GOTOWE ZA:", 10);
    LCD_PutChar('\n');
    LCD_PutString("Czas: 00:00", 11);

    // Wlacz przerwania globalne
    INTCON1bits.NSTDIS = 0;
}

// Pokazuje aktualny stan na wyswietlaczu
void pokaz_na_ekranie(void)
{
    char tekst[17];
    uint16_t minuty = czas_sekundy / 60;     // ile minut
    uint16_t sekundy = czas_sekundy % 60;    // ile sekund

    LCD_ClearScreen();
```

```
    if (stan == 0) {                    // zatrzymana
        if (czas_sekundy == 0) {
            if (skonczyl) {             // po zakonczeniu gotowania
                LCD_PutString("GOTOWE", 6);
            } else {                    // stan poczatkowy
                LCD_PutString("GOTOWE ZA:", 10);
            }
        } else {
            LCD_PutString("GOTOWE ZA:", 10);  // ustawiono czas, ale nie
wystartowano
        }
    } else if (stan == 1) {             // dziala
        LCD_PutString("Pracuje", 7);
        skonczyl = 1;
    } else if (stan == 2) {             // pauza
        LCD_PutString("Pauza", 5);
    }

    LCD_PutChar('\n');

    // Drugi wiersz
    if (stan == 0 && czas_sekundy == 0 && skonczyl) {
        LCD_PutString("SMACZNEGO!", 10);
    } else {
        // W trybie pauzy migaj dwukropkiem
        if (stan == 2 && migaj) {
            sprintf(tekst, "Czas: %02d %02d", minuty, sekundy);
        } else {
            sprintf(tekst, "Czas: %02d:%02d", minuty, sekundy);
        }
        LCD_PutString(tekst, 11);
    }
}
// Zatrzymanie odliczania
void zatrzymaj(void)
{
    stan = 0;                    // ustaw stan na zatrzymana
    odswiez_ekran = 1;          // odswiez ekran
}

// Zaczyna odliczanie
void zacznij(void)
{
    stan = 1;                           // stan - dziala
    ostatnia_sekunda = licznik_ms;      // zapamietaj czas startu
    odswiez_ekran = 1;                  // odswiez ekran
}
```