LABORATORY WORK NO. 3

PWM AND ADC

Subject: Microprocessors and their Programming
Student: Halil Ibrahim Bekli
Target Device: PIC16F1518

1. Goal and Task

Goal: To learn how to use the analog-to-digital converters (ADC) of the PIC16F1518 microcontroller (control registers ADCON0, ADCON1) and learn how to output a PWM signal.

Task: Create and connect a circuit according to the assigned variant and program the microcontroller so that the brightness of a connected LED changes depending on the voltage measured by the analog-to-digital converter (dependent on a potentiometer or thermistor).

2. Task Variant Data (Variant No. 19)

Based on the provided laboratory manual list, Variant 19 was selected with the following configuration:

- Analog Input: PORTA.1 (RA1/AN1) - Connected to Thermistor
- Output (PWM): PORTC.2 (RC2/CCP1) - Connected to LED
- Oscillator Frequency: 4 MHz (Internal)
- PWM Frequency: ~1 kHz (Calculated via PR2 register)

3. Circuit Diagram

The circuit is constructed based on the schematic provided in the laboratory manual:

1. Thermistor/Voltage Divider:
   - Resistor R4 connected to VDD (+5V).
   - Thermistor R2 connected to GND.
   - The junction point is connected to RA1 (Pin 3) for ADC reading.

2. LED Output:
   - An LED with a current-limiting resistor is connected to RC2 (Pin 13).
   - This pin corresponds to the CCP1 module output for PWM.

3. Power:
   - VDD (Pin 20) connected to +5V.
   - VSS (Pin 8/19) connected to GND.
   - MCLR (Pin 1) pulled high.

LABORATORY WORK NO. 3

PWM AND ADC

4. Code with Comments

```
/* * File:   LabWork3.c
 * Author: Halil Ibrahim Bekli
 *
 * Target: PIC16F1518
 * Description: ADC Reading from Thermistor (RA1) and controls LED brightness via PWM (RC2).
 * Based on Lab 3 requirements and Schematic.
 */

#include <xc.h>

// ========================================
// CONFIGURATION BITS
// ========================================
#pragma config FOSC = INTOSC    // Oscillator Selection (Internal Oscillator)
#pragma config WDTE = OFF       // Watchdog Timer Enable (Disabled)
#pragma config PWRTE = ON       // Power-up Timer Enable (Enabled)
#pragma config MCLRE = ON       // MCLR Pin Function Select (Enabled)
#pragma config CP = OFF         // Flash Program Memory Code Protection (Disabled)
#pragma config BOREN = OFF      // Brown-out Reset Enable (Disabled)
#pragma config CLKOUTEN = OFF   // Clock Out Enable (Disabled)
#pragma config IESO = OFF       // Internal/External Switchover (Disabled)
#pragma config FCMEN = OFF      // Fail-Safe Clock Monitor Enable (Disabled)

// CONFIG2
#pragma config WRT = OFF        // Flash Memory Self-Write Protection (Disabled)
#pragma config VCAPEN = OFF     // Voltage Regulator Capacitor Enable (Disabled)
#pragma config STVREN = ON      // Stack Overflow/Underflow Reset Enable (Enabled)
#pragma config LPBOR = OFF      // Low-Power Brown Out Reset (Disabled)
#pragma config LVP = OFF        // Low-Voltage Programming Enable (Disabled)

#define _XTAL_FREQ 4000000      // Fosc = 4MHz

// ========================================
// FUNCTIONS
// ========================================

void init_system() {
    // 1. Oscillator Setup (4MHz)
    OSCCONbits.IRCF = 0b1101;   // 4 MHz
    OSCCONbits.SCS = 0b10;      // Internal Oscillator Block

    // 2. Port Configuration
    // RA1 (AN1) is Input (Thermistor)
    TRISAbits.TRISA1 = 1;       // RA1 Input
    ANSELAbits.ANSA1 = 1;       // RA1 Analog

    // RC2 (CCP1) is Output (LED PWM)
    TRISCbits.TRISC2 = 0;       // RC2 Output
    ANSELCbits.ANSC2 = 0;       // RC2 Digital (Disable analog on this pin if present)

    // 3. ADC Configuration (per Lab instructions)
    // ADCON1: Right Justified, Fosc/16, Vref = VDD
    ADCON1bits.ADFM = 1;        // Right justified (10-bit result)
    ADCON1bits.ADCS = 0b101;    // Fosc/16
    ADCON1bits.ADPREF = 0b00;   // Vref+ is VDD

    // ADCON0: Select Channel AN1 (RA1)
    ADCON0bits.CHS = 0b00001;   // Select Channel AN1
    ADCON0bits.ADON = 1;        // Turn ADC On
```

LABORATORY WORK NO. 3

PWM AND ADC

```
    RC2 = 1;
}

void init_pwm() {
    // PWM Configuration based on Example 2 (2. PVZ)

    // 1. Set PWM Period
    // Formula: Period = (PR2 + 1) * 4 * Tosc * TMR2_Prescale
    // Target ~1kHz at 4MHz:
    // (255 + 1) * 4 * 0.25us * 4 = 1024us (~976Hz)
    PR2 = 255;

    // 2. Configure CCP1 Module for PWM
    // CCP1CON<3:0> = 1100 (PWM mode)
    CCP1CONbits.CCP1M = 0b1100;

    // 3. Initialize Duty Cycle to 0%
    CCPR1L = 0;
    CCP1CONbits.DC1B = 0;

    // 4. Configure and Start Timer2
    // T2CON: Prescaler 1:4, Timer On
    T2CONbits.T2CKPS = 0b01;    // Prescaler 1:4
    T2CONbits.TMR2ON = 1;       // Turn Timer2 On
}

// ========================================
// MAIN PROGRAM
// ========================================
int main() {
    unsigned int adc_result;    // 10-bit value (0-1023)

    init_system();
    init_pwm();

    while(1) {
        // --- Step 1: Read ADC (Thermistor) ---
        __delay_us(20);             // Acquisition delay
        ADCON0bits.GO = 1;          // Start conversion
        while(ADCON0bits.GO);       // Wait for conversion to complete

        // Combine High and Low bytes (10-bit result)
        adc_result = ((unsigned int)ADRESH << 8) + ADRESL;

        // --- Step 2: Update PWM Duty Cycle ---
        // We use the 10-bit ADC value directly for the 10-bit PWM duty cycle.
        // PWM Duty (10-bit) is split into CCPR1L (8 MSB) and DC1B (2 LSB).

        // Take the top 8 bits for CCPR1L
        CCPR1L = (adc_result >> 2);

        // Take the bottom 2 bits for CCP1CON<5:4> (DC1B)
        CCP1CONbits.DC1B = (adc_result & 0x03);

        // Note: As resistance changes -> Voltage changes -> PWM Duty changes -> LED brightness changes.
    }

    return 0;
}
```

LABORATORY WORK NO. 3

PWM AND ADC

5. Program Algorithm

1. Initialize System: Set Internal Oscillator to 4 MHz.
2. Configure GPIO: Set RA1 as Analog Input (for Thermistor) and RC2 as Digital Output (for LED/PWM).
3. Initialize ADC: Configure for Right Justified result, Fosc/16 clock, VDD reference, select Channel AN1.
4. Initialize PWM: Set PR2 for ~1kHz frequency, configure CCP1 in PWM mode.
5. Configure Timer2: Set Prescaler to 1:4 and enable Timer2.
6. Main Loop:
   a. Wait for acquisition time (20us).
   b. Start ADC conversion and wait for completion.
   c. Read 10-bit result from ADRESH:ADRESL.
   d. Write the upper 8 bits to CCPR1L and lower 2 bits to CCP1CON<5:4>.
   e. Repeat indefinitely.

6. Conclusions

In this laboratory work, the operation of the PIC16F1518 ADC and PWM modules was studied.
- We successfully configured the ADC module to read analog voltage from a thermistor on RA1.
- We configured the CCP module in PWM mode to generate a square wave on RC2.
- By mapping the ADC input directly to the PWM duty cycle, we achieved a system where LED brightness is directly proportional to the voltage.
- The use of registers ADCON0, ADCON1, CCP1CON, CCPR1L, and T2CON was practically demonstrated.