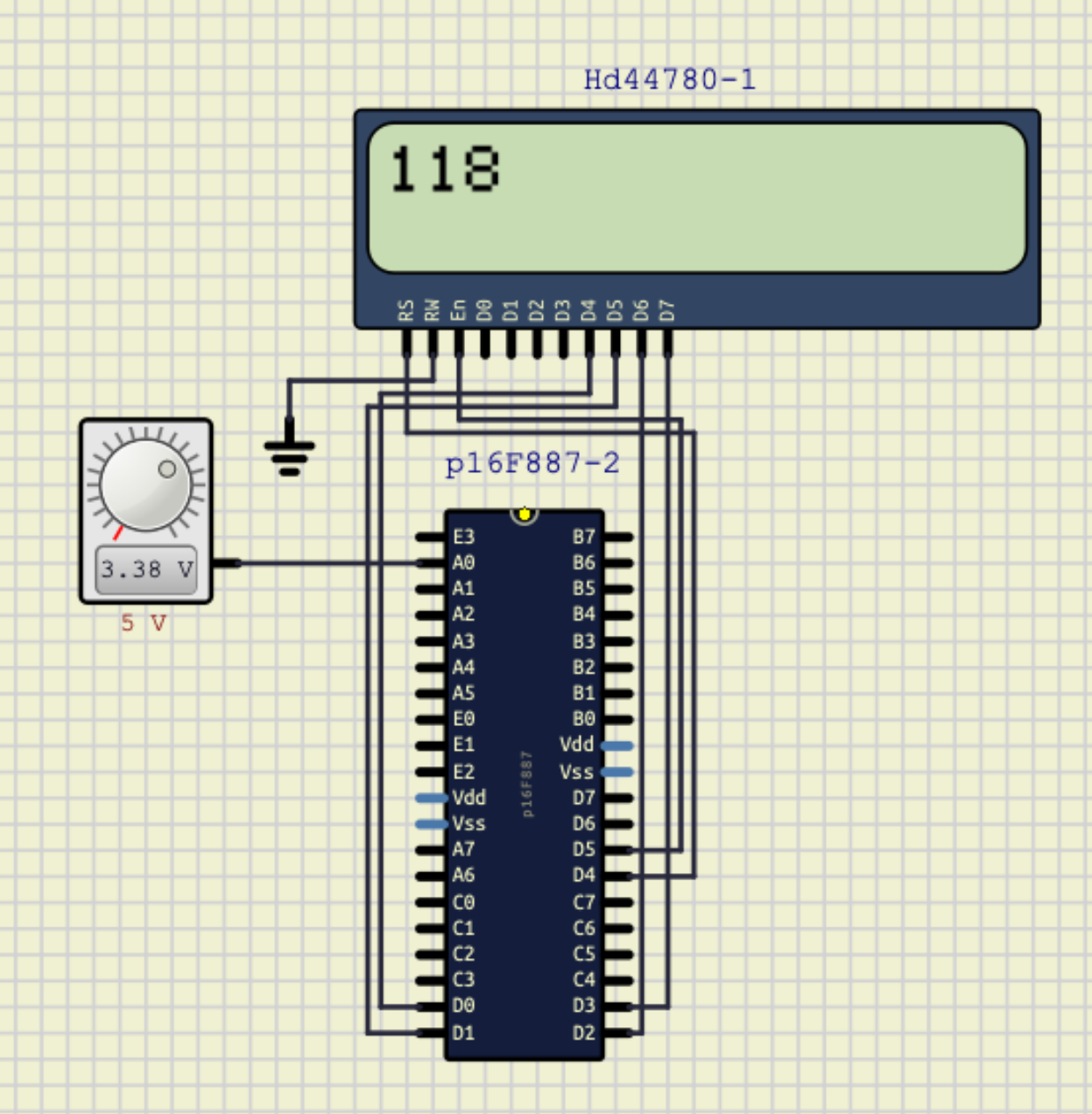
Assignment lab 4

Rashad Mohamed - 202200261

Objective:

In this Lab we are going to learn how to use the ADC and USART peripherals in the PIC microcontroller and integrate this knowledge in a practical example.

Circuit scheme:



CODE:

```
#define _XTAL_FREQ  4000000UL


 //#include "config_device.h"

 #include <xc.h>
```

```
#include <stdio.h>

#include <string.h>


typedef unsigned char uint8;

typedef unsigned short uint16;


#define SET_BIT(REG, BIT_POSN)      (REG |= (1 << BIT_POSN))

#define CLEAR_BIT(REG, BIT_POSN)    (REG &= ~(1 << BIT_POSN))

#define TOGGLE_BIT(REG, BIT_POSN)   (REG ^= (1 << BIT_POSN))

#define READ_BIT(REG, BIT_POSN)     ((REG >> BIT_POSN) & 1)


/****************** ADC Functions Start ******************/

void adc_initialize(void);

unsigned short adc_read(void);

/****************** ADC Functions End ******************/


/****************** UART Functions End ******************/

void uart_tx_initialize(void);

void uart_rx_initialize(void);

void uart_send(uint8 value);

uint8 uart_read(void);

/****************** UART Functions End ******************/


/****************** LCD Functions Start ******************/
```

```
void lcd_4bit_intialize(void);
void lcd_4bit_send_command(uint8 command);
void lcd_send_4bits(uint8 _data_command);
void lcd_4bit_send_enable_signal(void);
void lcd_4bit_send_char_data(uint8 data);
void lcd_4bit_set_cursor(uint8 row, uint8 coulmn);
void lcd_4bit_send_string(uint8 *str);
void lcd_4bit_clear(void);
void convert_uint16_to_string(uint16 value, uint8 *str);
void convert_uint8_to_string(uint8 value, uint8 *str);
/****************** LCD Functions End ******************/



uint16 adc_conversion_result = 0;
uint8 adc_output = 0;
uint8 adc_res_real_txt[7];

void main(void) {

  adc_initialize();
  uart_tx_initialize();
  lcd_4bit_intialize();
  lcd_4bit_clear();
```

```
  while (1) {
    adc_conversion_result = adc_read();
    adc_output = adc_conversion_result * 4.88f / 100;
    adc_output = adc_output * 3.6;
    uart_send(adc_output);
    convert_uint16_to_string(adc_output, adc_res_real_txt);
    lcd_4bit_set_cursor(1, 1);
    lcd_4bit_send_string(adc_res_real_txt);
  }

  return;
}

/****************** ADC Functions Start ******************/
void adc_initialize(void) {
   // 1. Enable Analog Pins.
   TRISA0 = 1; // Set Pin A0 as Input.

   ANSEL = 0x00; // Disable all Analog Pin Function.
   ANSELH = 0x00;
   // Enable Analog function to all required pins.
   SET_BIT(ANSEL, 0);

   //2. Configure the ADC module:
```

```
    //     * Select ADC conversion clock       (F/2)
    CLEAR_BIT(ADCON0, 7);
    CLEAR_BIT(ADCON0, 6);
    //     * Configure voltage reference       (Internal Vdd & Vss)
    CLEAR_BIT(ADCON1, 4);
    CLEAR_BIT(ADCON1, 5);
    //     * Select ADC input channel          (0000 = AN0)
    CLEAR_BIT(ADCON0, 5);
    CLEAR_BIT(ADCON0, 4);
    CLEAR_BIT(ADCON0, 3);
    CLEAR_BIT(ADCON0, 2);
    //     * Select result format              (Right justified ADFM=1)
    SET_BIT(ADCON1, 7);
    //     * Turn on ADC module
    SET_BIT(ADCON0, 0);
}

unsigned short adc_read(void) {
    unsigned short _adc_conversion_result = 0;
    //3. Wait the required acquisition time(2).
    __delay_us(20);
    //4. Start conversion by setting the GO/DONE bit.
    SET_BIT(ADCON0, 1);
    //5. Wait for ADC conversion to complete by Polling the GO/DONE.
```

```
    // The GO/DONE bit will remain 1 until the conversion is completed.
    while (READ_BIT(ADCON0, 1));
    //      * Waiting for the ADC interrupt (interrupts enabled)
    //6. Read ADC Result
    _adc_conversion_result = (double) (((ADRESH << 8) + ADRESL));
    return _adc_conversion_result;
}
/******************* ADC Functions End ******************/

/******************* UART Functions Start ******************/
void uart_tx_initialize(void) {
    TRISC6 = 0; // TX input pin.
    TRISC7 = 1; // RX input pin.
    //  Asynchronous Transmission Set-up:
    // Initialize the SPBRGH, SPBRG register pair and
    // the BRGH and BRG16 bits to achieve the desired baud rate.
    // desired baud rate = 9600
    SYNC = 0;
    BRG16 = 0;
    BRGH = 1;
    SPBRG = 25;
    //  Enable the asynchronous serial port by clearing
    //  the SYNC bit and setting the SPEN bit.
    SYNC = 0;
```

```
    SPEN = 1;


    // Enable the 9-bit transmission
    TX9 = 0;
    //  Enable the transmission by setting the TXEN
    //  control bit. This will cause the TXIF interrupt bit to be set.
    TXEN = 1;
}


void uart_rx_initialize(void) {
    TRISC6 = 0; // TX input pin.
    TRISC7 = 1; // RX input pin.
    //  Asynchronous Transmission Set-up:
    // Initialize the SPBRGH, SPBRG register pair and
    // the BRGH and BRG16 bits to achieve the desired baud rate.
    // desired baud rate = 9600
    SYNC = 0;
    BRG16 = 0;
    BRGH = 1;
    SPBRG = 25;
    //  Enable the asynchronous serial port by clearing
    //  the SYNC bit and setting the SPEN bit.
    SPEN = 1;
```

```
    // Enable the 9-bit transmission
    RX9 = 1;
    //  Enable the transmission by setting the TXEN
    //  control bit. This will cause the TXIF interrupt bit to be set.
    CREN = 1;

}

void uart_send(uint8 value) {
    TXREG = value; // Load the TX register.
    while (TXIF != 1); // wait till TX flag is set.
    TXIF = 0; // Clear the TX flag.
}

uint8 uart_read(void) {
    uint8 rx_value = 0;
    while (!RCIF);
    rx_value = RCREG;
    return rx_value;
}
/******************* UART Functions End *******************/

/******************* LCD Functions Start *******************/
void lcd_4bit_intialize(void) {
```

```
    // Initialize all connected pins
    TRISD0 = 0;
    TRISD1 = 0;
    TRISD2 = 0;
    TRISD3 = 0;
    TRISD4 = 0;
    TRISD5 = 0;

    __delay_ms(20);
    lcd_4bit_send_command(0x38);
    __delay_ms(5);
    lcd_4bit_send_command(0x38);
    __delay_us(150);
    lcd_4bit_send_command(0x38);

    lcd_4bit_send_command(0X01); // LCD_CLEAR
    lcd_4bit_send_command(0x02); // LCD_RETURN_HOME
    lcd_4bit_send_command(0x06); // LCD_ENTRY_MODE_INC_SHIFT_OFF
    lcd_4bit_send_command(0x0C); // LCD_DISPLAY_ON_UNDERLINE_OFF_CURSOR_OFF
    lcd_4bit_send_command(0x28); // LCD_4BIT_MODE_2_LINE
    lcd_4bit_send_command(0x80); // Cursor at beginning of fist line.
}

void lcd_4bit_send_command(uint8 command) {
```

```
    /* R/W Pin connected to the GND -> Logic (0) "Hard Wired" */
    /* Write Logic (0) to the "Register Select" Pin to select the "Instruction Register"  */
    PORTDbits.RD4 = 0;
    /* Send the Command through the (4-Pins" Data lines */
    lcd_send_4bits(command >> 4);
    /* Send the Enable Signal on the "E" Pin */
    lcd_4bit_send_enable_signal();
    /* Send the Command through the (4-Pins" Data lines */
    lcd_send_4bits(command);
    /* Send the Enable Signal on the "E" Pin */
    lcd_4bit_send_enable_signal();
}

void lcd_send_4bits(uint8 _data_command) {
    PORTDbits.RD0 = ((_data_command >> 0) & (uint8) 0x01);
    PORTDbits.RD1 = ((_data_command >> 1) & (uint8) 0x01);
    PORTDbits.RD2 = ((_data_command >> 2) & (uint8) 0x01);
    PORTDbits.RD3 = ((_data_command >> 3) & (uint8) 0x01);
}

void lcd_4bit_send_enable_signal(void) {
    PORTDbits.RD5 = 1;
    __delay_us(5);
    PORTDbits.RD5 = 0;
```

```
}

void lcd_4bit_send_char_data(uint8 data) {
    /* R/W Pin connected to the GND -> Logic (0) "Hard Wired" */
    /* Write Logic (1) to the "Register Select" Pin to select the "Data Register"  */
    PORTDbits.RD4 = 1;
    /* Send the Data through the (4-Pins" Data lines */
    lcd_send_4bits(data >> 4);
    /* Send the Enable Signal on the "E" Pin */
    lcd_4bit_send_enable_signal();
    /* Send the Data through the (4-Pins" Data lines */
    lcd_send_4bits(data);
    /* Send the Enable Signal on the "E" Pin */
    lcd_4bit_send_enable_signal();
}

void lcd_4bit_set_cursor(uint8 row, uint8 coulmn) {
    coulmn--;
    switch (row) {
        case 1: lcd_4bit_send_command((0x80 + coulmn));
            break;
        case 2: lcd_4bit_send_command((0xc0 + coulmn));
            break;
        default:;
```

```
    }
}

void lcd_4bit_send_string(uint8 *str) {
    while (*str) {
        lcd_4bit_send_char_data(*str++);
    }
}

void lcd_4bit_clear(void) {
    lcd_4bit_send_command(0X01); // LCD_CLEAR
}

void convert_uint16_to_string(uint16 value, uint8 *str) {
    uint8 Temp_String[6] = {0};
    uint8 DataCounter = 0;
    memset(str, ' ', 5);
    str[5] = '\0';
    sprintf((char *) Temp_String, "%i", value);
    while (Temp_String[DataCounter] != '\0') {
        str[DataCounter] = Temp_String[DataCounter];
        DataCounter++;
    }
}
```

```
void convert_uint8_to_string(uint8 value, uint8 *str) {
    memset((char *) str, '\0', 4);
    sprintf(str, "%i", value);
}
```