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:

A screenshot of a computer

Description automatically generated

### 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);

}