LAB 5 RAPORT

Creator: Sampo Kajaste 1881563

SISÄLTÖ

1 GOAL

In this lab, you will learn how to use A/D converter to acquire data, manipulate it, and display the result on LCD.

1.1 Objectives

Upon completion of this lab, you will be able to:

• Describe the characteristics of A/D converter.

• Configure internal A/D of PIC18. • Implement data manipulation techniques.

• Display characters on LCD.

1.2 A/D Converter

So far, we have used PIC18 for digital signals. While we want to benefit from the advantages that digital systems can offer us, we should consider that most of real world variables are analog in nature such as temperature, sound, weight ... These variables can take an infinite range of different values. Therefore, it is necessary that a microcontroller be able to read values that are analog, process them, and generate the result on the output devices such as LCD. To convert a signal from analog to digital form, we use Analog to Digital (A/D) converter. The circuit of an A/D is complex. However, A/Ds are available as ready-to-use chips or modules inside microcontrollers. PIC18F87J11 has a 10-bit A/D with 15 input channels. For details of A/D and its configurations, refer to chapter 21 of PIC18F87J11 datasheet. In this lab, you use A/D to display room temperature on LCD. PIC18 explorer board has an on board temperature sensor: MPC9701A. MPC9701A converts temperature in Celsius to voltage. The output voltage of the sensor can be connected to the A/D directly without any additional signal-conditioning circuit. The transfer function for MPC9701A is: Vout=400mv+T×19.5m, where Vout is voltage generated by sensor and T is temperature in Celsius. For details of MPC9701A, please refer to the datasheet of the sensor. Implementation of the transfer function in assembly is challenging since it requires arithmetic operations for floating-point numbers. Hence, in this lab, you will write a C-program instead of assembly.

1.3 LCD

The PIC board has an LCD with two lines and 16 characters (2×16 LCD). The LCD is connected to the PIC through an SPI I/O expander: MCP23S17. MCP23S17 receives commands and data from microcontroller for LCD through SPI port. Then, it converts serial bits into bytes and send 2 them to the LCD. The course website has a sample program for LCD. Figure 5.1 shows the main() function of the program. Before sending any data to the LCD, the main() function initializes the LCD through LCDInit(). Then, it configures SPI port to communicate with MCP23S17. Inside the while() loop, "Hello PIC18" is sent to the LCD. void main(void) { // Initialize the LCD display LCDInit(); TXSTA = 0b10100100; SPBRG = 0xff; RCSTA = 0b10010000; while(1) { // Write the command to start on line 1 LCDLine_1(); // Write the data one char at a time. d_write('H'); d_write('e'); d_write('l'); d_write('l'); d_write('0'); // Write the command to start on line 2 LCDLine_2(); // Write the data to line 2 one char at a time d_write('P'); d_write('I'); d_write('C'); d_write('1'); d_write('8'); delay_1s();

} } Figure 5.1 Sample program for LCD In PIC Explorer board, temperature sensor is connected to RA1. Configure RA1 as input to the A/D. Write a C program that generates interrupt when A/D finishes the conversion. Inside the interrupt service routine, read digital value of temperature from A/D and send it to the LCD.

2 CODE

```
;ADCON0 setup
 ;VCFG1 = 0
 ;VCFG0 = 0
 ;CHS<3:0> = 0001
 ;GO/DONE = 0
 ;ADON = 1
 ;ADCON1 setup
 ;ADFM = 1
 ;ADCAL = 0
 ;ACQT<2:0> =  011
 ;ADCS<2:0> = 001

 ;ANCON0 = 0x05
 ;ANCON1 = 0x99

 ; vout = 19.5mV*celcius+V0(dec celsius)
 ;(vout = 400m
 ;19.5/3.125 = 6.24
 ;400/3.125 = 128
 ; 6*advalue + 128 <--- tää laskee oikein

 ;3.2/1024 = 0.003125
 ;advaleu/1024 = vin/3.2
```

```
 ;-40 +125'C  == 165 väli
 ;v0/3.2 = 40/165 = v0=0.78
 ;0.78/0.195 = 39.8
*/

// PIC18F87J11 Configuration Bit Settings

// 'C' source line config statements

// CONFIG1L
#pragma config WDTEN = OFF      // Watchdog Timer Enable bit (WDT disabled (control is placed on SWDTEN bit))
#pragma config STVREN = OFF     // Stack Overflow/Underflow Reset Enable bit (Reset on stack overflow/underflow disabled)
#pragma config XINST = OFF      // Extended Instruction Set Enable bit (Instruction set extension and Indexed Addressing mode disabled (Legacy mode))

// CONFIG1H
#pragma config CP0 = OFF        // Code Protection bit (Program memory is not code-protected)

// CONFIG2L
#pragma config FOSC = HS        // Oscillator Selection bits (HS oscillator)
#pragma config FCMEN = ON       // Fail-Safe Clock Monitor Enable bit (Fail-Safe Clock Monitor enabled)
#pragma config IESO = ON        // Two-Speed Start-up (Internal/External Oscillator Switchover) Control bit (Two-Speed Start-up enabled)

// CONFIG2H
#pragma config WDTPS = 32768    // Watchdog Timer Postscaler Select bits (1:32768)
```

```
// CONFIG3L

#pragma config EASHFT = ON      // External Address Bus Shift Enable bit (Address shifting enabled, address on external bus is offset to start at 000000h)

#pragma config MODE = MM        // External Memory Bus Configuration bits (Microcontroller mode - External bus disabled)

#pragma config BW = 16          // Data Bus Width Select bit (16-bit external bus mode)

#pragma config WAIT = OFF       // External Bus Wait Enable bit (Wait states on the external bus are disabled)


// CONFIG3H

#pragma config CCP2MX = DEFAULT // ECCP2 MUX bit (ECCP2/P2A is multiplexed with RC1)

#pragma config ECCPMX = DEFAULT // ECCPx MUX bit (ECCP1 outputs (P1B/P1C) are multiplexed with RE6 and RE5; ECCP3 outputs (P3B/P3C) are multiplexed with RE4 and RE3)

#pragma config PMPMX = DEFAULT  // PMP Pin Multiplex bit (PMP port pins connected to EMB (PORTD and PORTE))

#pragma config MSSPMSK = MSK7   // MSSP Address Masking Mode Select bit (7-Bit Address Masking mode enable)


// #pragma config statements should precede project file includes.

// Use project enums instead of #define for ON and OFF.


#include <xc.h>

#ifndef __PIC18LCD_C

#define __PIC18LCD_C


#include <p18F87J11.h>


#define     LCD_CS
            (LATAbits.LATA2)                    //LCD chip select

#define     LCD_CS_TRIS                                     (TRISAbits.TRISA2)
            //LCD chip select
```

```
#define     LCD_RST
            (LATFbits.LATF6)                    //LCD chip select
#define     LCD_RST_TRIS                                (TRISFbits.TRISF6)
            //LCD chip select

#define LCD_TXSTA_TRMT          (TXSTAbits.TRMT)
#define LCD_SPI_IF                                  (PIR1bits.SSPIF)
#define LCD_SCK_TRIS                        (TRISCbits.TRISC3)
#define LCD_SDO_TRIS                        (TRISCbits.TRISC5)
#define LCD_SSPBUF                                  (SSPBUF)
#define LCD_SPICON1                                 (SSP1CON1)
#define LCD_SPICON1bits         (SSP1CON1bits)
#define LCD_SPICON2                                 (SSP1CON2)
#define LCD_SPISTAT                                 (SSP1STAT)
#define LCD_SPISTATbits         (SSP1STATbits)

//extern void Delay(void);
void Delay(void)
{
            char Dreg1;
            char Dreg2;

            Dreg1=255;
            Dreg2=255;

            Nop();
            Nop();
            Nop();
            Nop();
```

```
                while(Dreg1>0)
                        {
                        while(Dreg2>0)
                                {
                                Dreg2--;
                                }
                        Dreg1--;
                        }

return;
}

//extern void SDelay(void);
void SDelay(void)
{

                char Dreg1;
                char Dreg2;

                Dreg1=255;
                Dreg2=255;

                while(Dreg1>0)
                        {
                        while(Dreg2>0)
                                {
                                Dreg2--;
                                }
                        Dreg1--;
```

```
                }


return;
}

void delay_1s(void)
{
        char Dreg1;
        char Dreg2;
        char Dreg3;

        Dreg1=255;
        Dreg2=255;
        Dreg3=5;

        while(Dreg1>0)
                {
                while(Dreg2>0)
                        {
                        while(Dreg3>0)
                                {
                                Dreg3--;
                                }

                        Dreg2--;
                        }
                Dreg1--;
                }
```

```
return;
}

#pragma code

//*****************************************************************
// LCD busy delay
//*****************************************************************
void LCDBusy(void)
{
		SDelay();
		SDelay();
}
//*****************************************************************
// Write to MCP923S17 Port A
//*****************************************************************
void WritePortA(char b)
{
		LCD_CS = 0;

		LCD_SSPBUF = 0x40;
		while(!LCD_SPI_IF);
		LCD_SPI_IF = 0;

		LCD_SSPBUF = 0x12;
		while(!LCD_SPI_IF);
		LCD_SPI_IF = 0;
```

```
		LCD_SSPBUF = b;
		while(!LCD_SPI_IF);
		LCD_SPI_IF = 0;

		LCD_CS = 1;
}
//*****************************************************************
// Write to MCP923S17 Port B
//*****************************************************************
void WritePortB(char b)
{
		LCD_CS = 0;

		LCD_SSPBUF = 0x40;
		while(!LCD_SPI_IF);
		LCD_SPI_IF = 0;

		LCD_SSPBUF = 0x13;
		while(!LCD_SPI_IF);
		LCD_SPI_IF = 0;

		LCD_SSPBUF = b;
		while(!LCD_SPI_IF);
		LCD_SPI_IF = 0;

		LCD_CS = 1;
}
//*****************************************************************
// Write the data to the display
```

```
//*********************************************************************
void d_write(char b)
{
		WritePortA(0x80);
		LCDBusy();
		WritePortB(b);
		Nop();
		Nop();
		Nop();
		Nop();
		WritePortA(0xC0);
		Nop();
		Nop();
		Nop();
		Nop();
		Nop();
		Nop();
		WritePortA(0x00);
		TXREG = b;
                                        //carriage return
		while(!LCD_TXSTA_TRMT);                 //wait for data TX
		LCD_TXSTA_TRMT = 0;


		return;
}
//*********************************************************************
// Send a instruction to the display
//*********************************************************************
void i_write(char b)
```

```
{
		WritePortA(0x00);
		LCDBusy();
		WritePortB(b);
		Nop();
		Nop();
		Nop();
		Nop();
		WritePortA(0x40);
		Nop();
		Nop();
		Nop();
		Nop();
		Nop();
		Nop();
		WritePortA(0x00);
}
//*********************************************************************
// Write to line 1 of the display
//*********************************************************************
void LCDLine_1(void)
{
		i_write(0x80);
}
//*********************************************************************
// Write to line 1 of the display
//*********************************************************************
void LCDLine_2(void)
{
```

```
		i_write(0xC0);
}
//*********************************************************************
// To clear the display
//*********************************************************************
void LCDClear(void)
{
		i_write(0x01);
}
//*********************************************************************
// Function to write to the PORT
//*********************************************************************
void InitWrite(char b)
{
		WritePortA(0);
		WritePortB(b);
		Nop();
		Nop();
		Nop();
		WritePortA(0x40);
		Nop();
		Nop();
		Nop();
		Nop();
		Nop();
		Nop();
		Nop();
		Nop();
		Nop();
		Nop();
```

```
		WritePortA(0);
}
//*****************************************************************
// Initialize MCP923S17 Port A
//*****************************************************************
void InitPortA_SPI(char b)
{
		LCD_CS = 0;
		LCD_SSPBUF = 0x40;
		while(!LCD_SPI_IF);
		LCD_SPI_IF = 0;

		LCD_SSPBUF = 0x00;
		while(!LCD_SPI_IF);
		LCD_SPI_IF = 0;

		LCD_SSPBUF = b;
		while(!LCD_SPI_IF);
		LCD_SPI_IF = 0;

		LCD_CS = 1;
}
//*****************************************************************
// Initialize MCP923S17 Port B
//*****************************************************************
void InitPortB_SPI(char b)
{
		LCD_CS = 0;
		LCD_SSPBUF = 0x40;
```

```
		while(!LCD_SPI_IF);
		LCD_SPI_IF = 0;

		LCD_SSPBUF = 0x01;
		while(!LCD_SPI_IF);
		LCD_SPI_IF = 0;

		LCD_SSPBUF = b;
		while(!LCD_SPI_IF);
		LCD_SPI_IF = 0;

		LCD_CS = 1;
}
//*****************************************************************
// Initialize MCP923S17 SPI
//*****************************************************************
void InitSPI(void)
{
		LCD_SCK_TRIS = 0;
		LCD_SDO_TRIS = 0;

		LCD_SPICON1 = 0x22;
		LCD_SPISTATbits.CKE = 1;
		//LCD_SPISTATbits.SMP = 0;
		LCD_SPI_IF = 0;
}
//*****************************************************************
// LCD Initialization function
//*****************************************************************
```

```
void LCDInit(void)
{
        LCD_CS_TRIS = 0;
        LCD_CS = 1;
        Delay();
        Delay();
        Delay();

        LCD_RST_TRIS = 0;
        LCD_RST = 0;
        Delay();
        LCD_RST = 1;

        InitSPI();
        InitPortA_SPI(0);
        InitPortB_SPI(0);

        WritePortA(0);

        Delay();
        InitWrite(0x3C);                    //0011NFxx

        Delay();
        InitWrite(0x0C);                    //Display
Off

        Delay();
        InitWrite(0x01);                    //Display
Clear
```

```
                Delay();
                InitWrite(0x06);                                    //Entry
mode
}

#endif
void interrupt isr(){
    int value;
    int temp;
    value = ADRESH<<8 + ADRESL;
    temp = 6 * value + 0x80;

    LCDLine_1();.
    d_write(temp);
    LCDLine_2();
    d_write('C');
    d_write('E');
    d_write('L');
    d_write('C');
    d_write('I');
    d_write('U');
    d_write('S');
    delay_1s();
    PIR1bits.ADIF = 0;
}
void main(void) {
    int value;
    float temp;
```

```
    // Initialize the LCD display
    LCDInit();
    TXSTA = 0b10100100;
    SPBRG = 0xff;
    RCSTA = 0b10010000;

    TRISAbits.RA1 = 1;
    PIR1bits.ADIF = 0;
    PIE1bits.ADIE = 0;
    INTCONbits.PEIE = 1;
    INTCONbits.GIE = 1;

    ADCON0 =0x05;
    ADCON1 =0x99;



    while(1){
       Delay();
       ADCON0bits.GO = 1;
    }
    return;
}
```

3 SELF-REFLECT

I know that this code wont work. I didn't understand how to send the value to LCD correctly.

I think I got how the ADC is calculated and How to set up the bits corresponding to ADC and Interrupt. But I have no previous knowledge regarding writing C. Some of the syntaxes are foreign to me. Thus making it hard to change the values to chars.