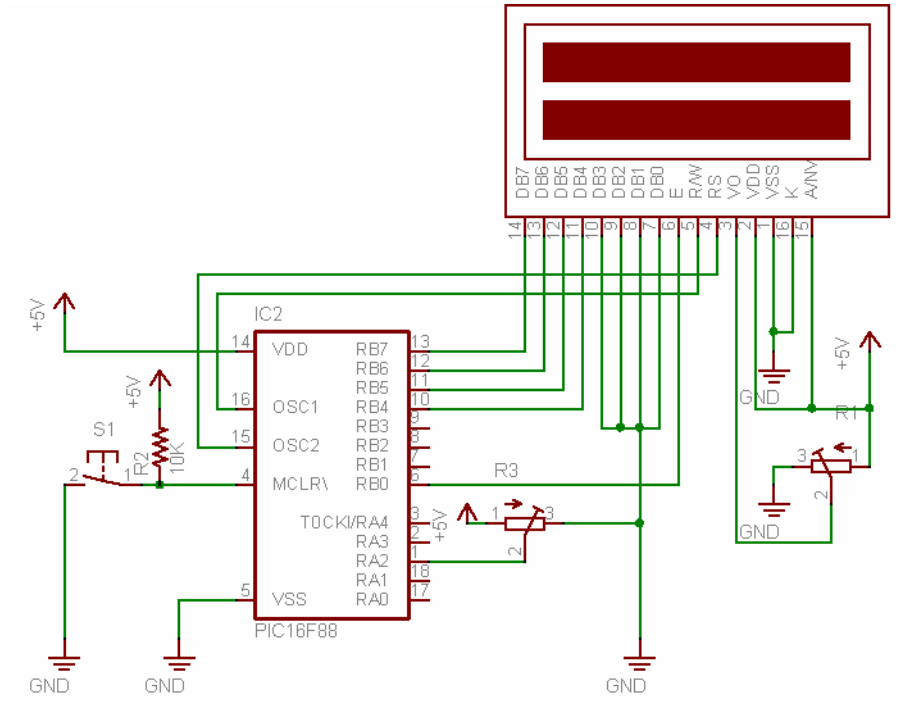
Project 8 ADC with LCD Display

โจทย์

สร้างวงจรและเขียนโปรแกรมสำหรับไมโครคอนโทรลเลอร์ PIC16F88 เพื่อแปลงสัญญาณอะนาล็อกเป็นข้อมูลแบบดิจิตอล แล้วแสดงข้อมูลบน LCD Display ประเภท Character ขนาด 16x2 ตัวอักษร โดยให้สร้างสัญญาณอะนาล็อกจากตัวต้านทานแบบปรับค่าได้เพื่อให้โปรแกรมแสดงผลค่าแรงดันไฟเมื่อหมุนตัวต้านทานไปยังตำแหน่งต่างๆ

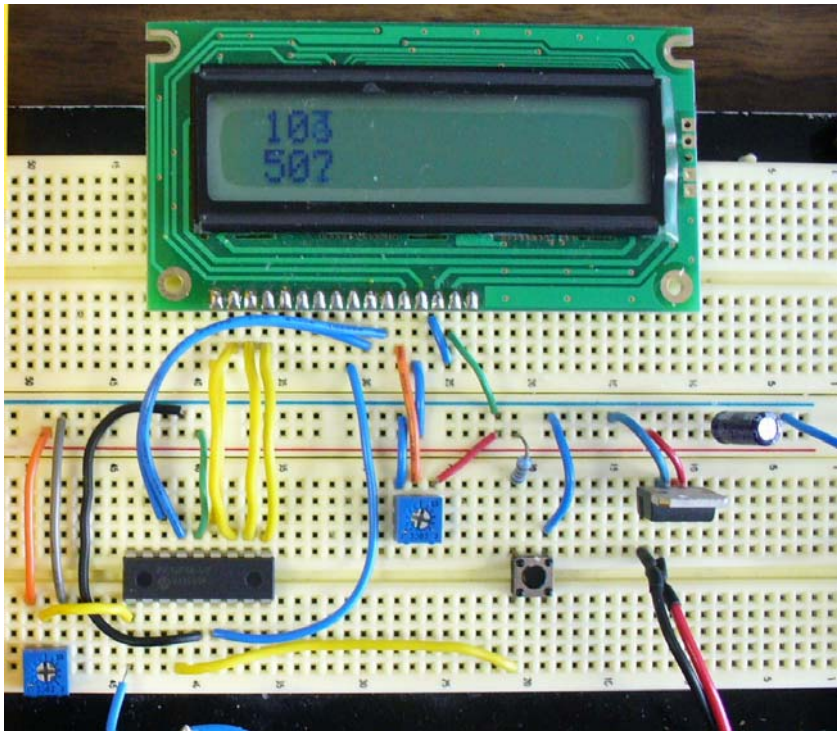
ขั้นตอนการปฏิบัติ

1) การออกแบบวงจร



รูปที่ 1 แผนผังวงจรสำหรับแสดงผลการใช้งาน A/D Module ผ่านทาง LCD

2) การต่อวงจร



รูปที่ 2 รูปแสดงการต่อวงจร

3) การเขียนโปรแกรม

เนื่องจาก PIC16F88 มีวงจรแปลงสัญญาณแบบอะนาล็อกเป็นสัญญาณดิจิตอล (ADC : analog-to-digital converter)ในตัวอยู่แล้วถึง 7 ช่องสัญญาณจึงไม่จำเป็นต้องใช้ไอซีเพิ่มเติมแต่อย่างใด (สามารถดูข้อมูลได้ใน datasheet หน้า 113 หัวข้อ 12.0 Analog-To-Digital Converter (A/D) Module) เพียงแต่เขียนโปรแกรมเพื่อใช้งานวงจรที่มีภายในให้ถูกต้องก็สามารถใช้งานได้ ดังเช่นโปรเจกที่ 7 เราจะต้องทำการกำหนดค่าเกี่ยวกับ oscillator และ analog input ก่อน จากนั้นเป็นส่วน LCD ซึ่งในโปรเจกนี้มีการเปลี่ยนตำแหน่งพอร์ทที่ใช้ควบคุม RS, RW, EN เป็น RA6, RA7 และ RB0 ตามลำดับจึงต้องมีการเปลี่ยนค่าที่ define ไว้ใน lcd4bit.h ด้วย ถัดมาจึงเป็นส่วนของ ADC ซึ่งการใช้งานจะมีรีจีสเตอร์ที่จำเป็นต้องกำหนดค่าก่อนใช้งานดังนี้

ANSEL = 0x04 ใช้เลือกชนิดของสัญญาณให้เป็นอะนาล็อกหรือดิจิตอลซึ่งเคยกำหนดไปแล้วก่อนส่วนของ LCD หากกำหนดไว้ถูกต้องแล้วก็ไม่จำเป็นต้องกำหนดอีกครั้ง หากยังไม่ถูกต้องเราสามารถกำหนดเป็นรายช่องสัญญาณได้ เช่นในโปรเจกนี้เราต้องการใช้ช่องสัญญาณที่ 2 (RA2/AN2) ให้กำหนดให้ ANS2 = 1 และที่สำคัญคือจะต้องกำหนดพอร์ทให้เป็นอินพุทโดยกำหนดให้ TRISA2 = 1 ด้วยจึงจะใช้งานได้อย่างถูกต้อง

ADCON1 = 0x91 ซึ่งมีรายละเอียดการตั้งค่าบิตที่สำคัญดังต่อไปนี้

ADFM = 1 เนื่องจาก ADC ให้ผลลัพธ์มีความละเอียดที่ 10 bit จะต้องใช้หน่วยความจำขนาด 2 Byte ในการเก็บค่า ซึ่งจะมีรีจีสเตอร์สำหรับเก็บผลลัพธ์โดยเฉพาะชื่อ ADRESH และ ADRESL เราจึงกำหนดให้ค่าผลลัพธ์ที่ได้อยู่ชิดขวา ทำให้ 6 bit ซ้ายของ ADRESH มีค่าเป็น 0 เพื่อให้ง่ายในการใช้งานต่อไป

ADCS2 = 0 กำหนดให้สัญญาณนาฬิกาสำหรับกระบวนการแปลงสัญญาณมีค่าเท่ากับสัญญาณนาฬิกาของระบบ

VCFG<1:0> = 00 กำหนดให้ใช้ VDD เป็น VREF+ และ ให้ใช้ VSS เป็น VREF-

ADCON0 = 0x80 ซึ่งมีรายละเอียดการตั้งค่าบิตที่สำคัญดังต่อไปนี้

ADCS<1:0> = 10 กำหนดความเร็วสัญญาณนาฬิกาสำหรับกระบวนการแปลงสัญญาณ = FOSC/32 (เนื่องจาก ADCS2 = 0)

CHS<2:0> = 010 เลือกช่องสัญญาณที่ต้องการแปลงค่าเป็นช่องที่ 2

GODONE = 0 ใช้สำหรับเริ่มกระบวนการแปลงสัญญาณ โดยกำหนดให้มีค่าเป็น 1 เมื่อต้องการเริ่มใช้งาน

ADON = 1 เปิดการใช้งาน A/D converter Module

เมื่อกำหนดรีจิสเตอร์ต่างๆแล้วมีลำดับวิธีการใช้งานดังนี้

1. ADCON0 = 0x80; ในกรณีที่ต้องการเลือกช่องสัญญาณก่อนจะแปลงค่าให้เลือกช่องสัญญาณก่อน
2. DelayUs(20); จะต้องรอให้สัญญาณของช่องสัญญาณที่เลือกไว้คงที่ก่อนซึ่งเวลาที่ใช้นี้ขึ้นกับค่าความต้านทานภายในของต้นกำเนิดสัญญาณ ระยะเวลา 20 us ถือว่านานมากเพราะคิดจากความต้านทานภายในที่ 10 K ohm โดยคำนวณจากสมการ Acquisition Time ในหัวข้อ 12.1 Acquisition Requirements หน้า 117
3. GODONE = 1; เริ่มกระบวนการแปลงสัญญาณ
4. while(GODONE); วนเช็คค่า GODONE หากมีค่าเป็น 0 แสดงว่าแปลงค่าเสร็จสมบูรณ์แล้ว
5. temp = ((unsigned int)ADRESH<<8) | (unsigned int)ADRESL; นำผลลัพธ์ที่ได้ออกมาจากรีจิสเตอร์ ADRESH และ ADRESL โดยเก็บไว้ในตัวแปลชนิด unsigned int หรือ long (เนื่องจากสามารถรับข้อมูลได้ถึง 16 bit)

ในส่วนการนำข้อมูลที่ได้ไปแสดงผลยัง LCD นั้น เราจะต้องทำการแปลงค่าที่เป็นตัวเลขเป็นรูปแบบตัวอักษรเสียก่อนโดยใช้ฟังก์ชัน sprintf(char *buf, const char * fmt, ...) ทำให้เราต้อง include stdio.h ไว้ด้วย ดูตัวอย่างได้จาก source code ของโปรแกรม

โปรแกรม project08a.c

```
#include<pic.h>
#include<stdio.h>
#include "lcd4bit.h"
#include "delay.h"
__CONFIG(UNPROTECT & LVPDIS & BOREN & MCLREN & PWRTDIS & WDTDIS & INTIO );

void main (void)
{
   char lcdtext[17];        // Tempolary array of charracter for display on LCD.
   unsigned int temp;       // Tempolary variable for keep ADC result.
   unsigned int temp2;      // Tempolary variable for calculate result back to mV unit.
   IRCF2   =  1;
   IRCF1 =  1;
   IRCF0 =  1;              // IRCF<2:0> = 111: internal RC oscillator frequency = 8 MHz
   SCS1  =  0;
   SCS0  =  0;              // SCS<1:0> = 00: Oscillator mode defined by FOSC<2:0>
   while(!IOFS);            // Wait for the frequency to become stable
   ANSEL = 0x00;            // Select every bit of port A as digital I/O
   lcd_init(1);             // Initialize LCD use high bite of port

   TRISA2  =  1;            // Set port RA2 as input.
   ANS2  =  1;              // Set port RA2 as analog input.
   ADCON0  =  0x91;         // Fosc/32, select analog channel 2, Enable A/D module
   ADCON1  =  0x80;         // Right justified VREF+ = VDD, VREF- = VSS

   while(1)
   {
      DelayUs(20);                        // Wait the required acquisition time (~20 us).
      GODONE = 1;                         // Start acquiring.
      while(GODONE);                      // Wait until the ADC module finish.

      temp = ((unsigned int)ADRESH<<8) | (unsigned int) ADRESL;
      // Obtain the ADC reading.

      temp2 = (unsigned int)((((unsigned long) temp)*5000)/1024);
      // Convert ADC value to millivolt.
```

```
      sprintf(lcdtext,"%4d",temp);// Convert integer value to string.
      lcd_display_line1(lcdtext);        // Display value (raw data).
      sprintf(lcdtext,"%4d",temp2);      // Convert integer value to string.
      lcd_display_line2(lcdtext);        // Display value (in millivolt unit).   }
}
```

คำอธิบายโปรแกรม

โปรแกรม project08a.c มีการประกาศและกำหนดค่าในส่วนต้นเหมือนกับโปรแกรม project07.c ต่อมาเป็นการกำหนดค่ารีจิสเตอร์สำหรับ A/D Module จากนั้นจึงจะเข้า main loop ซึ่งทำหน้าที่สั่งเริ่มการแปลงสัญญาณอะนาล็อกเป็นดิจิตอล, แปลงข้อมูลดิบที่ได้ให้อยู่ในหน่วย mV และ แปลงข้อมูลชนิดเลขจำนวนเต็มให้เป็นข้อมูลตัวหนังสือเพื่อแสดงผลออกทาง LCD ต่อไป

4) การเขียนโปรแกรมโดยใช้งาน Interrupt

การใช้งาน interrupt ทำให้ไม่ต้องรอกระบวนการแปลงสัญญาณด้วยการวนเช็คการเปลี่ยนค่าสัญญาณจากรีจิสเตอร์ GODONE อีก ทำให้สามารถนำเวลาส่วนนี้มาใช้งานอื่นๆได้ หากต้องการใช้งานจะต้องมีการกำหนดรีจิสเตอร์เพิ่มเติมได้แก่

ADIF = 0 ล้างสัญญาณแสดงการเสร็จสิ้นกระบวนการแปลงสัญญาณอะนาล็อกเป็นดิจิตอล (ในครั้งแรกก่อนใช้งาน)

ADIE = 1 เปิดใช้งานสัญญาณ interrupt จาก A/D Module

PEIE = 1 เปิดใช้งานสัญญาณ interrupt จากกลุ่ม peripheral

GIE = 1 เปิดใช้งานสัญญาณ interrupt หลัก

สำหรับวิธีการใช้งานให้ทำกระบวนการที่ 1.-3. ในครั้งแรกก่อนเข้า main loop เมื่อ A/D Module ทำการแปลงสัญญาณเสร็จเรียบร้อยก็จะส่ง interrupt flag: ADIF = 1 โดยอัตโนมัติ เราจึงต้องสร้างฟังก์ชันสำหรับจัดการกับ interrupt ที่เกิดขึ้น โดยตรวจสอบจากรีจิสเตอร์ ADIF หากมีค่าเป็น 1 จึงนำผลลัพธ์ที่ได้ออกมาจากรีจิสเตอร์ ADRESH และ ADRESL แล้วจึงกำหนดให้ ADIF เป็น 0 เพื่อไว้ใช้ตรวจสอบในครั้งต่อไป และทำกระบวนการที่ 1.-3. อีกครั้ง เป็นอันเสร็จสิ้นการจัดการกับ interrupt ในส่วน main loop จึงสามารถทำงานใดๆต่อได้ รวมถึงการแสดงผลลัพธ์ที่ได้จาก A/D Module ออกทาง LCD จนกว่าจะมีสัญญาณ interrupt ใหม่เกิดขึ้น สามารถดูตัวอย่าง source code สำหรับการใช้งาน interrupt จากโปรแกรม project08b.c

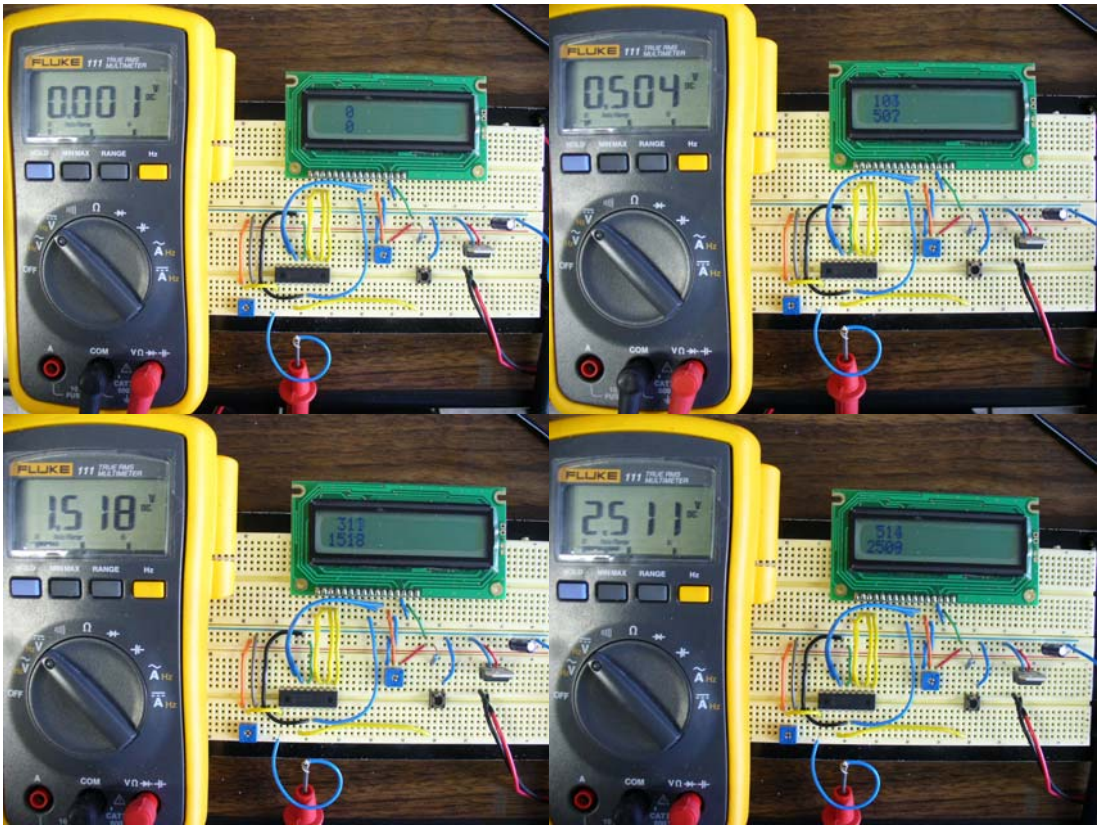
โปรแกรม project08b.c

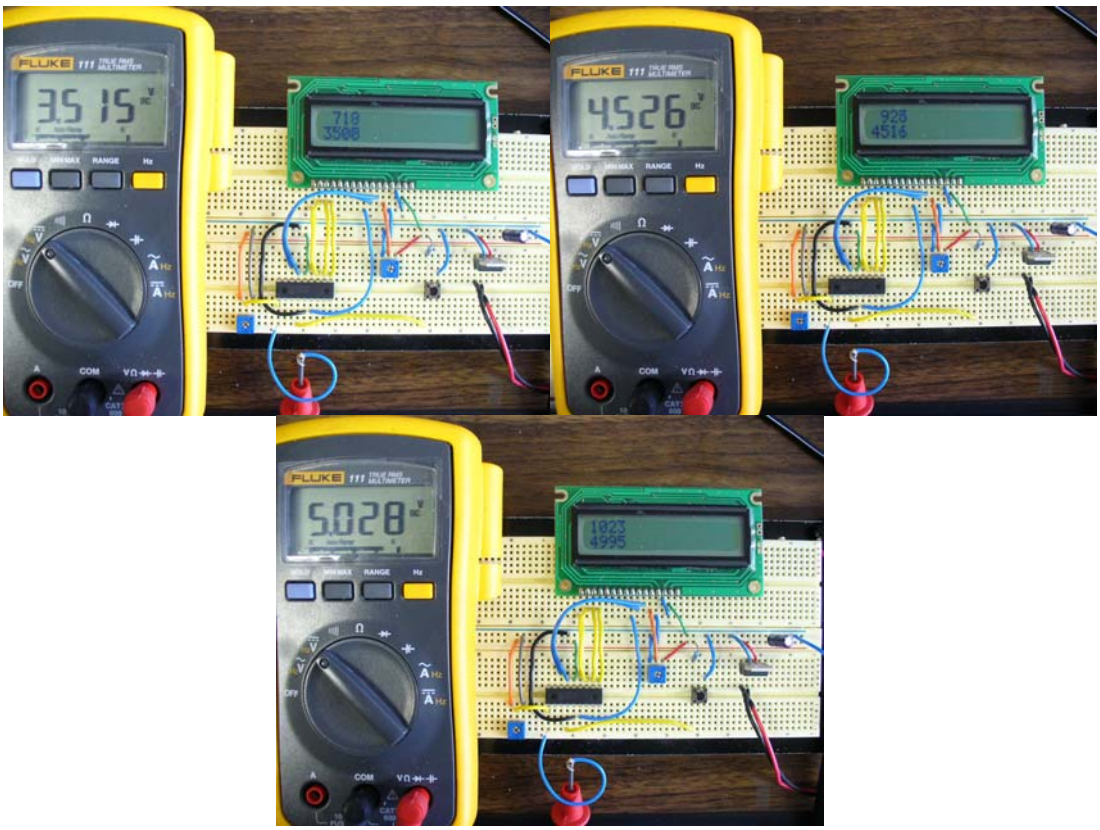
```
#include<pic.h>
#include<stdio.h>
#include "lcd4bit.h"
#include "delay.h"
__CONFIG(UNPROTECT & LVPDIS & BOREN & MCLREN & PWRTDIS & WDTDIS & INTIO );

unsigned int temp;
unsigned int temp2;

void main (void)
{
   char res[17];
   IRCF2   =  1;
   IRCF1 =  1;
   IRCF0 =  1;              // IRCF<2:0> = 111: internal RC oscillator frequency = 8 MHz
   SCS1  =  0;
   SCS0  =  0;              // SCS<1:0> = 00: Oscillator mode defined by FOSC<2:0>
   while(!IOFS);            // Wait for the frequency to become stable
   ANSEL = 0x00;            // Select every bit of port A as digital I/O
   lcd_init(1);             // Initialize LCD use high bite of port

   TRISA2  =  1;            // Set port RA2 as input.
   ANS2  =  1;              // Set port RA2 as analog input.
   ADCON0  =  0x91;         // Fosc/32, select analog channel 2, Enable A/D module
   ADCON1  =  0x80;         // Right justified Vref+ = VDD, Vref- = VSS
   ADIF  =  0;              // Clear A/D interrupt flag at first time.
   ADIE  =  1;              // Enable A/D interrupt signal.
   PEIE  =  1;              // Enable Peripheral interrupt.
   GIE     =  1;            // Enable global interrupt.
```

```
   DelayUs(20);              // Wait the required acquisition time (~20 us).
   GODONE = 1;               // Start acquiring.

   while(1)
   {
      sprintf(res,"%4d",temp);                  // Convert integer value to string.
      lcd_display_line1(res);                   // Display value (in millivolt unit).
      sprintf(res,"%4d",temp2);                 // Convert integer value to string.
      lcd_display_line2(res);                   // Display value (in millivolt unit).
   }
}
static void interrupt isr(void)
{
   if(ADIF)
   {
      temp = ((unsigned int)ADRESH<<8) | (unsigned int)ADRESL;      // Read result first.
      ADIF = 0;                            // Clear interrupt flag bit.
      temp2 = (int)((((unsigned long) temp)*5000)/1024);            // Convert to mV unit.
      DelayUs(20);                         // Wait the required acquisition time (~20 us).
      GODONE = 1;                          // Start acquiring.
   }
}
```

5) การทดสอบการทำงาน

สามารถทดลองได้โดยนำมัลติมิเตอร์มาวัดแรงดันไฟฟ้าจากตัวต้านทานแบบปรับค่าได้ แล้วนำมาเทียบกับผลที่แสดงออกทาง LCD ว่ามีค่าใกล้เคียงกันหรือไม่ เช่นผลการทดลองดังรูปที่ 3





รูปที่ 3 รูปแสดงผลการทดลอง