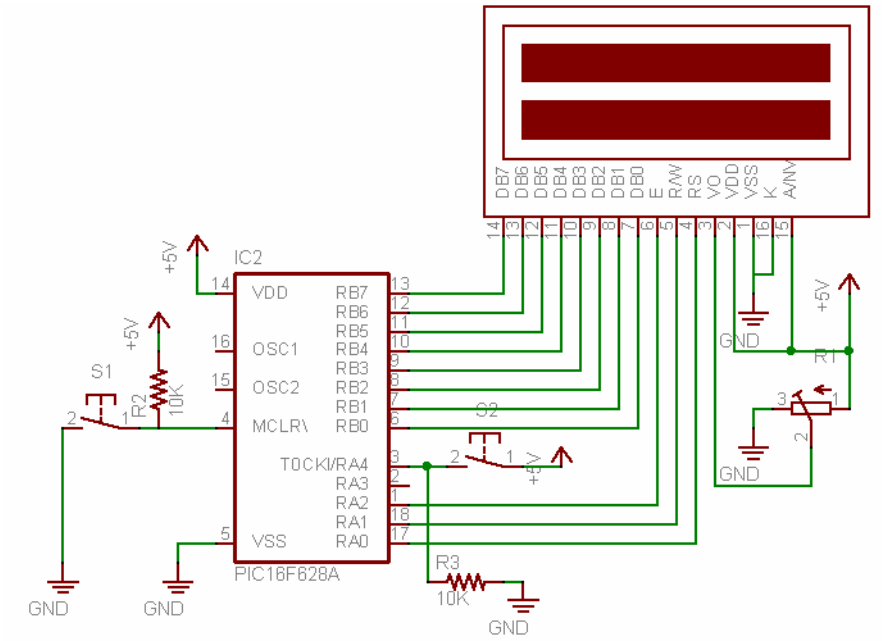
Project 6 LCD

โจทย์

สร้างวงจรและเขียนโปรแกรมสำหรับแสดงข้อความบน LCD Display ประเภท Character ขนาด 16x2 ตัวอักษร โดยใช้สวิตช์แบบกดติดปล่อยดับช่วยในการอินพุทสัญญาณเพื่อทดสอบคำสั่งแสดงผลแบบต่างๆ 4-5 คำสั่ง เช่น การแสดงผลในบรรทัดที่ 1, การแสดงผลในบรรทัดที่ 2, การล้างหน้าจอ, การเลื่อนตำแหน่ง curser เป็นต้น

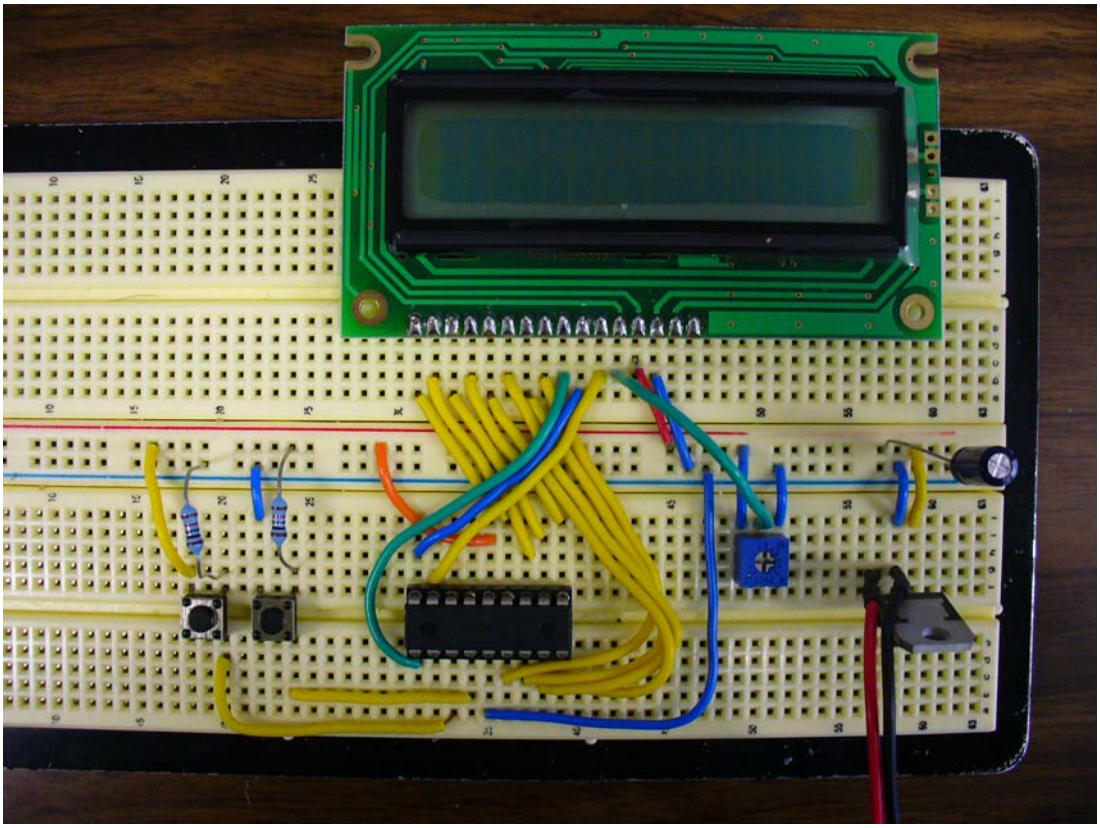
ขั้นตอนการปฏิบัติ

1) การออกแบบวงจร



รูปที่ 1 แผนผังวงจรควบคุมการแสดงข้อความบน LCD โดยใช้สัญญาณข้อมูล 8 บิต

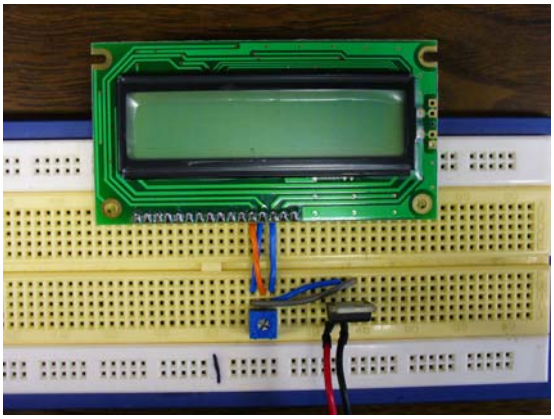
2) การต่อวงจร

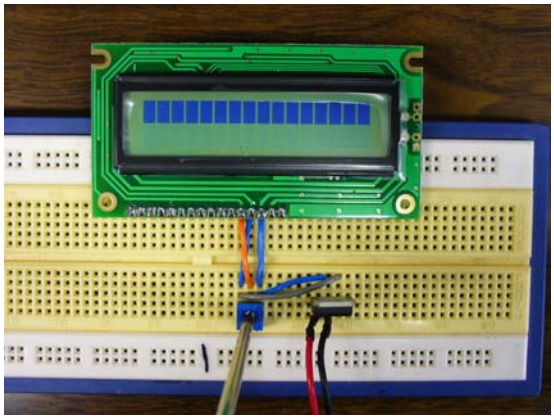


รูปที่ 2 รูปแสดงการต่อวงจร

3) การทดสอบ LCD

การทดสอบการทำงานของ LCD ว่าปกติหรือไม่ทำได้โดย ต่อสัญญาณไฟฟ้าไปที่ขา Vss Vdd และ Vo ดังรูปที่ 1 โดยที่ Vo นั้นให้ปรับตัวต้านทานแบบปรับค่าได้จน LCD แสดงแต่ละ pixel เป็นสีทึบเต็มช่องดังตัวอย่างในรูปที่ 3





รูปที่ 3 รูปแสดงการทดสอบ LCD ก่อนปรับแรงดัน **รูปที่ 4 รูปแสดงการทดสอบ LCD เมื่อจ่ายแรงดันที่เหมาะสม**

หากไม่สามารถปรับให้เห็นได้ LCD อาจจะชำรุดและไม่สามารถแสดงข้อความตามที่เราเขียนโปรแกรมไว้

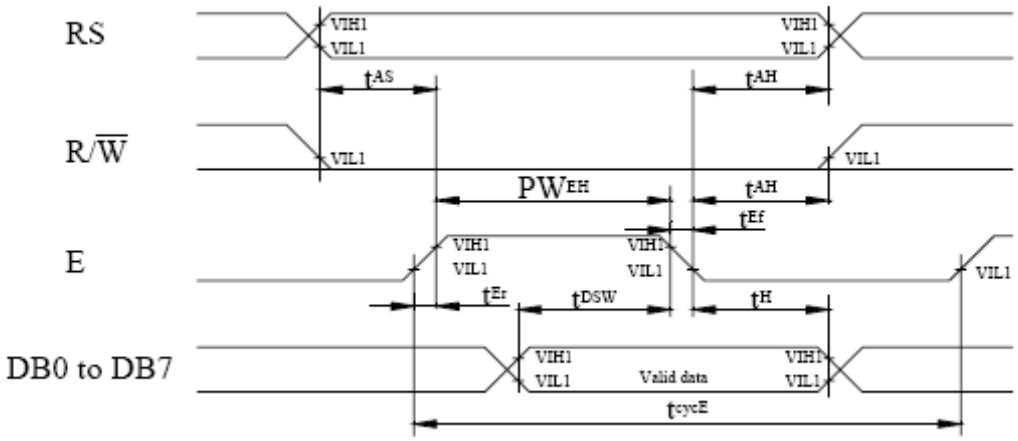
4) การเขียนโปรแกรม

การเขียนโปรแกรมติดต่อกับ LCD นั้นควรกำหนด port สำหรับส่วนข้อมูลขนาด 8 bit ให้โดยเฉพาะเพราะจะทำให้สะดวกและรวดเร็ว (สามารถส่งข้อมูลทั้ง 8 bit ได้ภายใน 2 คำสั่ง) แต่ PIC 16F628A นั้นมี I/O port เพียง 2 port คือ A และ B แต่ RA5

นั้นไม่สามารถใช้เป็น output port เพื่อส่งข้อมูลไปยัง LCD ได้ ดังนั้นจึงต้องเลือกใช้ port B และเพื่อให้ง่ายในการเขียนโปรแกรม เราอาจจะประกาศตัวแปรที่เราเข้าใจง่ายเป็น port B เช่น " **#define lcd_DATA PORTB** " และ " **#define TRISlcd_DATA TRISB** "

นอกจาก port สำหรับส่งข้อมูลแล้ว ยังจะต้องมีสัญญาณควบคุมการทำงานของ LCD อีก 3 สัญญาณคือ RS, R/W และ E เพื่อให้ง่ายในการเขียนโปรแกรมเราอาจจะประกาศตัวแปรที่เราเข้าใจง่ายเพื่อใช้อ้างถึง เช่น " **#define lcd_RS RA0** " เมื่อเราใช้ RA0 เป็นขาควบคุมสัญญาณ RS, " **#define lcd_RW RA1** " เมื่อเราใช้ RA1 เป็นขาควบคุมสัญญาณ R/W และ " **#define lcd_EN RA2** " เมื่อเราใช้ RA2 เป็นขาควบคุมสัญญาณ E ทั้งนี้การจะใช้ port A เป็น I/O สำหรับ output สัญญาณควบคุม จะต้องกำหนดให้ register **CMCON** = 0x07 เสียก่อน (ดูรายละเอียดได้จาก data sheet หัวข้อที่ 5 I/O PORTS หน้า 29)

การเขียนโปรแกรมสำหรับควบคุมการแสดงผลของ LCD นั้นไม่ยาก แต่มีรายละเอียดค่อนข้างมาก สิ่งสำคัญคือต้องศึกษาขั้นตอนและลำดับเวลาในการทำงาน (Timing characteristics) ของ LCD แต่ละรุ่นจาก data sheet โดยทั่วไปแล้วการติดต่อกับ LCD มักจะเป็นการติดต่อเพื่อเขียนข้อมูลลงหน่วยความจำ (Write Operation) เพื่อให้ LCD แสดงผลตามที่ต้องการ หรืออย่างไรก็ตามเราสามารถอ่านข้อมูลจากหน่วยความจำใน LCD ได้ด้วยหากต้องการทราบว่าขณะนั้น LCD กำลังแสดงผลข้อความอะไรอยู่ หรือ LCD พร้อมต่อการรับคำสั่งต่อไปหรือไม่ด้วยการอ่านค่า busy flag จากขาที่ 14 (DB7) ของ LCD สำหรับ LCD ที่ใช้ใน project นี้มีลำดับเวลาการทำงานสำหรับการเขียนข้อมูลลงบน LCD ดังนี้



จากรูปแสดงให้เห็นว่าจะต้องเขียนโปรแกรมเพื่อควบคุมขา RS และ R/W ให้คงสัญญาณไว้จนกว่าจะทำการเขียนข้อมูลเสร็จ และต้องรอเวลาอย่างน้อย tAS-tEr จากนั้นจึงเปิดการทำงานของขา E (E=1) และคงสัญญาณไว้เป็นเวลาอย่างน้อย tDSW จากเวลาที่เริ่มส่งข้อมูล และคงข้อมูลนี้ไว้เป็นเวลาอย่างน้อย tH หลังจากปิดการทำงานของขา E (E=0)

สรุปลำดับการทำงานเพื่อแสดงข้อความบน LCD ได้ดังนี้

1. กำหนดค่า TRISlcd_DATA = 0x00ff, RS=0, RW=1, E=1 เพื่ออ่านค่า busy flag
2. รอจนกว่า LCD จะพร้อมรับคำสั่งใหม่ (busy flag DB7=0)
3. ปิดการทำงานของขา E พร้อมทั้งกำหนด TRISlcd_DATA = 0x00 (ให้ port B เป็น output)
4. กำหนดค่า RS=1, RW=0, E=1 เพื่อเขียนตัวอักษรไปยัง LCD (RS=0 เพื่อส่งคำสั่งควบคุม)
5. ส่งรหัส ASCI ของตัวอักษรที่ต้องการแสดงไปยัง DB0-DB7 เช่น ต้องการส่งตัวอักษร "A" แสดงบน LCD ให้ lcd_DATA = 'A' (PORTB='A') หรือส่งคำสั่งควบคุมอื่นๆ (ขึ้นอยู่กับค่า RS ตามข้อ 4) เช่น lcd_DATA = 0x01 เพื่อล้างหน้าจอแสดงผล

หมายเหตุ การส่งรหัส ASCI ผ่าน DB0-DB7 เป็นการส่งคำสั่งให้ LCD ดึงภาพที่เก็บอยู่ในหน่วยความจำ CGRAM มาใช้ตามที่อยู่ DB0-DB7 ซึ่งจัดเรียงตัวอักษรไว้ตามรหัส ASCI อยู่แล้ว ส่วนคำสั่งที่ใช้ควบคุมอื่นๆ สามารถดูได้จาก Instruction table ใน data sheet ของ LCD

6. ปิดการทำงานของขา E
7. ถ้าต้องการส่งข้อความที่มีหลายตัวอักษรให้ทำซ้ำขั้นตอนที่ 1-5

เนื่องจากการเขียนโปรแกรมสำหรับควบ LCD มีรายละเอียดมากและต้องเขียนโค้ดยาว เพื่อให้ง่ายในการนำไปใช้งานจึงควรเขียนโปรแกรมเฉพาะในส่วนของการควบคุม LCD ไว้เป็น header file ซึ่งจะประกอบด้วยฟังก์ชันที่สามารถเรียกไปใช้ได้ทันทีจาก main program หากต้องการใช้งาน LCD จึงเพิ่มเข้าไว้ใน project ดังนี้

โปรแกรม project06.c

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

void main (void)
{
   char text1[] = "Hello World";
   char text2[] = "Text 2";
   char text3[] = "Str @ L1 C5";
   int show=1;
   CMCON=0x07;                          //Set port A as output
   TRISA4=1;                            //Set RA4 as input
   lcd_init();                          //Initialise LCD

   while(1)
   {
      if(RA4==1)                        //Press the button or not
      {
         while(RA4==1){}                //Wait untill release the button
         switch(show)
         {
            case 1    :      lcd_display_line1(text1); break;  //Display text1 at line 1
            case 2    :      lcd_display_line2(text2); break;  //Display text2 at line 2
            case 3    :      lcd_clear(); break;               //Clear display
            case 4    :      lcd_display_both(text2,text1); break;
                             /*Display text2 at line 1 & text1 at line2*/
            case 5    :      lcd_clear();
                             lcd_cursor(1,5);            //Move cusor to line 1 column 5
                             lcd_display(text3); break;  //Display text3 at present cursor
         }
         if (show<5) show++; else show=1;                  //Go to next command
      }
   }
}
```

Header file lcd.h

```
#ifndef __lcd_h_
#define __lcd_h_

#define lcd_DATA PORTB
#define TRISlcd_DATA TRISB

#define lcd_RS      RA0
#define TRISlcd_RS TRISA0

#define lcd_RW      RA1
#define TRISlcd_RW TRISA1

#define lcd_EN      RA2
#define TRISlcd_EN TRISA2

#define lcd_BF      RB7
#define TRISlcd_BF TRISB7

#define input 1                  //Status of I/O port as input
#define output 0                 //Status of I/O port as output
```

```
#define lcd_clear() lcd_command(0x01)
#define lcd_origin() lcd_command(0x02)
#define lcd_cursor_right() lcd_command(0x14)
#define lcd_cursor_left() lcd_command(0x10)

// Function prototypes
void lcd_init(void);
void lcd_wait(void);
void lcd_display(const unsigned char*);
void lcd_display_line1(const unsigned char*);
void lcd_display_line2(const unsigned char*);
void lcd_display_both(const unsigned char*, const unsigned char*);
void lcd_cursor(unsigned char, unsigned char);
void lcd_command(unsigned char);

// Function for initial LCD
void lcd_init()
{
   TRISlcd_DATA = 0x00;
   TRISlcd_RS = output;
   TRISlcd_RW = output;
   TRISlcd_EN = output;

   lcd_EN = 0;

   lcd_command(0x38);                                    // 8 bit mode, 2 lines.
   lcd_command(0x0c);                                    // Display, no cursor.
   //lcd_command(0x0e);                                  // Display, show cursor.
   lcd_clear();
   lcd_origin();
}

// Function for wait LCD ready
void lcd_wait()
{
   do
   {
      TRISlcd_DATA = 0x00ff;
      lcd_RS = 0;
      lcd_RW = 1;
      lcd_EN = 1;
   }
   while(lcd_BF);                                        // Check Busy Flag LCD.

   lcd_EN = 0;                                           // Disable LCD.
   TRISlcd_DATA = 0x0000;
}

// Function for display text on LCD
void lcd_display(const unsigned char* text)
{
   while(*text)
   {
      lcd_wait();                                        // Wait LCD ready
      lcd_RS = 1;                                        // Select data register.
      lcd_RW = 0;                                        // Write to the LCD register.
      lcd_EN = 1;                                        // Enable LCD.
      lcd_DATA = *text;                                  // Send data to LCD.
      lcd_EN = 0;                                        // Disable LCD.
      text++;
   }
}

void lcd_display_both(const unsigned char* text1, const unsigned char* text2)
{
   lcd_clear();
   lcd_origin();
   lcd_display(text1);
   lcd_cursor(2,1);
   lcd_display(text2);
}

void lcd_display_line1(const unsigned char* text1)
{
   unsigned char no = 0;
   lcd_origin();

   while(*text1)
```

```
   {
      lcd_wait();                                          // Wait LCD ready
      lcd_RS = 1;                                          // Select data register.
      lcd_RW = 0;                                          // Write to the LCD register.
      lcd_EN = 1;                                          // Enable LCD.
      lcd_DATA = *text1;                                   // Send data to LCD.
      lcd_EN = 0;                                          // Disable LCD.
      text1++;
      no++;
   }
   while(no <= 16)
   {
      lcd_display(" ");
      no++;
   }
}

void lcd_display_line2(const unsigned char* text2)
{
   unsigned char no = 0;
   lcd_cursor(2,1);

   while(*text2)
   {
      lcd_wait();                                          // Wait LCD ready
      lcd_RS = 1;                                          // Select data register.
      lcd_RW = 0;                                          // Write to the LCD register.
      lcd_EN = 1;                                          // Enable LCD.
      lcd_DATA = *text2;                                   // Send data to LCD.
      lcd_EN = 0;                                          // Disable LCD.
      text2++;
      no++;
   }
   while(no <= 16)
   {
      lcd_display(" ");
      no++;
   }
}

// Function for move cursor of LCD.
void lcd_cursor(unsigned char line, unsigned char no)
{
   switch(line)
   {
      case 1:        switch(no)
                {
                       case 1:  lcd_command(0x80); break;  //Move cursor to line 1 column 1
                       case 2:  lcd_command(0x81); break;  //Move cursor to line 1 column 2
                       case 3:  lcd_command(0x82); break;
                       case 4:  lcd_command(0x83); break;
                       case 5:  lcd_command(0x84); break;
                       case 6:  lcd_command(0x85); break;
                       case 7:  lcd_command(0x86); break;
                       case 8:  lcd_command(0x87); break;
                       case 9:  lcd_command(0x88); break;
                       case 10: lcd_command(0x89); break;
                       case 11: lcd_command(0x8a); break;
                       case 12: lcd_command(0x8b); break;
                       case 13: lcd_command(0x8c); break;
                       case 14: lcd_command(0x8d); break;
                       case 15: lcd_command(0x8e); break;
                       case 16: lcd_command(0x8f); break;
                       default: break;
                }
                break;

      case 2:        switch(no)
                {
                       case 1:  lcd_command(0xc0); break;  //Move cursor to line 2 column 1
                       case 2:  lcd_command(0xc1); break;  //Move cursor to line 2 column 2
                       case 3:  lcd_command(0xc2); break;
                       case 4:  lcd_command(0xc3); break;
                       case 5:  lcd_command(0xc4); break;
                       case 6:  lcd_command(0xc5); break;
                       case 7:  lcd_command(0xc6); break;
                       case 8:  lcd_command(0xc7); break;
                       case 9:  lcd_command(0xc8); break;
                       case 10: lcd_command(0xc9); break;
```

```
                    case 11: lcd_command(0xca); break;
                    case 12: lcd_command(0xcb); break;
                    case 13: lcd_command(0xcc); break;
                    case 14: lcd_command(0xcd); break;
                    case 15: lcd_command(0xce); break;
                    case 16: lcd_command(0xcf); break;
                    default: break;
                }
                break;

      default:break;
   }
}

// Function for send a command to LCD
void lcd_command(unsigned char command)
{
   lcd_wait();                                  // Wait LCD ready
   lcd_RS = 0;                                  // Select command register.
   lcd_RW = 0;                                  // Write to the LCD register.
   lcd_EN = 1;                                  // Enable LCD.
   lcd_DATA = command;                          // Send data to LCD.
   lcd_EN = 0;                                  // Disable LCD.
}

#endif
```

คำอธิบายโปรแกรม

จะเห็นได้ว่าใน project นี้ประกอบด้วยโปรแกรมหลักและ header file ทำให้ code ในโปรแกรมหลักสั้นและเข้าใจง่าย คือมีเพียงการประกาศตัวแปรของคำที่ต้องการแสดงผล การสั่งงานต่างๆ จะเรียกผ่านฟังก์ชันที่มีใน lcd.h โดยก่อนจะใช้ฟังก์ชันใดๆ จะต้องเรียกฟังก์ชัน lcd_init() เสียก่อน เนื่องจากฟังก์ชันนี้จะเป็นตัวกำหนดพารามิเตอร์ต่างๆ ของ LCD ให้พร้อมใช้งาน

ในส่วน lcd.h มีส่วนต่างๆ ที่สำคัญดังนี้

ส่วนการประกาศการใช้งานของ port ต่างๆ อยู่ที่ส่วนบนสุด เราสามารถเปลี่ยนแปลง port ที่ใช้ติดต่อกับ LCD ได้ในส่วนนี้ โดยไม่จำเป็นต้องเปลี่ยนแปลงโปรแกรมภายในแต่อย่างใด สามารถนำ header file นี้ไปใช้กับ PIC รุ่นอื่นได้ง่ายเพียงแค่ทำการเปลี่ยน port ที่ต้องการใช้เท่านั้น (จะต้องศึกษา PIC แต่ละรุ่นให้ดีว่าสามารถใช้ port ไดได้บ้าง) สำหรับ PIC ที่ใช้ใน project นี้ไม่สามารถใช้ port A เป็น output ได้ทุกบิต ดังนั้นจึงใช้ port B สำหรับรับ-ส่ง ข้อมูล หรือ instruction code และใช้ port A0-A2 เป็น port สำหรับควบคุม RS, RW และ E ตามลำดับ

ส่วนการประกาศชื่อฟังก์ชันต่างๆ และพารามิเตอร์ เราสามารถเรียกใช้ฟังก์ชันต่างที่ได้ประกาศไว้ ดังนี้

ฟังก์ชัน	คำอธิบาย
void lcd_init(void);	กำหนดพารามิเตอร์ต่างๆ ของ LCD
void lcd_wait(void);	ตรวจสอบว่า LCD พร้อมรับคำสั่งใหม่หรือยัง จาก busy flag (ผู้ใช้ไม่จำเป็นต้องใช้ฟังก์ชันนี้)
void lcd_display(const unsigned char*);	แสดงข้อความออก LCD โดยเริ่มจากตำแหน่ง curser ปัจจุบัน
void lcd_display_line1(const unsigned char*);	แสดงข้อความออก LCD ในบรรทัดที่ 1
void lcd_display_line2(const unsigned char*);	แสดงข้อความออก LCD ในบรรทัดที่ 2
void lcd_display_both(const unsigned char*, const unsigned char*);	แสดงข้อความออก LCD ในบรรทัดที่ 1 และ 2 ตามลำดับ
void lcd_cursor(unsigned char, unsigned char);	เลื่อน curser ไปยังตำแหน่ง บรรทัด และ คอลัมน์ ตามลำดับ (เป็นการเรียกใช้ฟังก์ชัน lcd_command()

	อีกทีหนึ่ง)
void lcd_command(unsigned char);	ส่งคำสั่งควบคุม LCD

นอกจากนี้ยังมีการประกาศคำสั่งควบคุม LCD ที่ใช้งานบ่อย ผ่านฟังกชัน lcd_command(unsigned char); ดังนี้

ชื่อที่ใช้แทนฟังก์ชัน	ฟังก์ชัน lcd_command()	คำอธิบาย
lcd_clear()	lcd_command(0x01)	ล้างหน้าจอแสดงผล
lcd_origin()	lcd_command(0x02)	เลื่อน curser ไปยังจุดเริ่มต้น
lcd_cursor_right()	lcd_command(0x14)	เลื่อน curser ไปทางขวา
lcd_cursor_left()	lcd_command(0x10)	เลื่อน curser ไปทางซ้าย

รูปแบบการทำงานที่สำคัญของฟังก์ชัน lcd_wait เป็น read operation ของ LCD ดังนี้

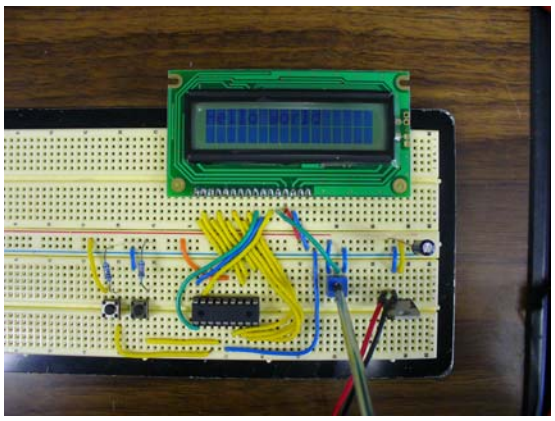
```
do
{
        TRISlcd_DATA = 0x00ff;   เพื่อกำหนดให้ I/O port เป็น input port ชั่วคราว
        lcd_RW = 1;              เพื่ออ่านค่าจาก LCD ผ่าน lcd_DATA
        lcd_RS = 0;              เพื่อเลือกข้อมูลในส่วนควบคุม LCD
        lcd_EN = 1;              เพื่อส่งสัญญาณค่างต้นไปยัง LCD
}
while(lcd_BF);                   วนรอจนกว่า LCD จะพร้อมทำงาน โดยอ่านค่าจาก busy flag (DB7)
lcd_EN = 0;                      ยกเลิกการส่งสัญญาณไปยัง LCD
TRISlcd_DATA = 0x0000;           เพื่อกำหนดให้ I/O port กลับเป็น output port
```

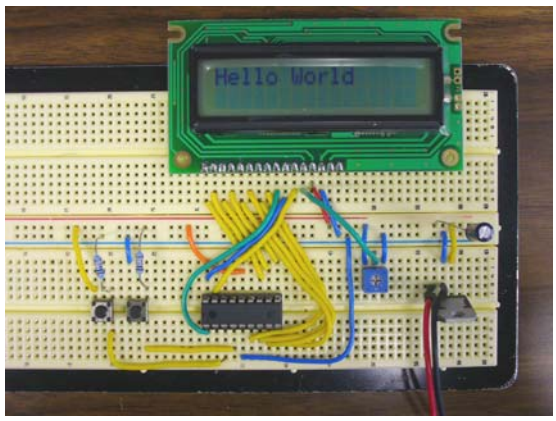
รูปแบบการทำงานที่สำคัญของฟังก์ชันกลุ่ม lcd_display lcd_command เป็น write operation ของ LCD ดังนี้

```
lcd_wait();                      เพื่อตรวจสอบให้แน่ใจว่า LCD พร้อมรับคำสั่ง
lcd_RS = 0 หรือ 1;               เลือกว่าจะส่งคำสั่งควบคุม หรือส่งข้อความ
lcd_RW = 0;                      เพื่อส่งค่าไปยัง LCD
lcd_EN = 1;                      เพื่อส่งสัญญาณค่างต้นไปยัง LCD
lcd_DATA = command หรือ text;    ส่งคำสั่งหรือข้อความที่ต้องการ
lcd_EN = 0;                      ยกเลิกการส่งสัญญาณไปยัง LCD
```

5) การทดสอบการทำงาน

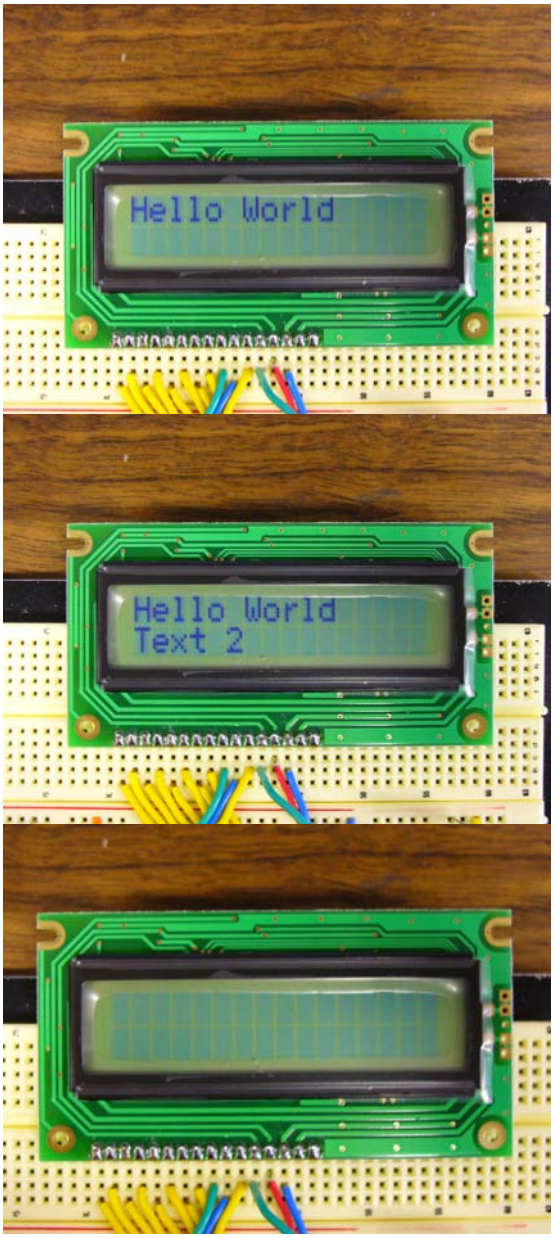
จากโปรแกรมหากเรากดปุ่มเพื่ออินพุทสัญญาณจากขา RA4 LCD ควรจะแสดงข้อความ “Hello world” จากการปรับค่าแรงดันจากตัวต้านทานในขั้นตอนทดสอบ LCD อาจทำให้ภาพที่ปรากฏไม่ชัดเจน ดังรูปที่ ให้ทำการปรับตัวต้านทานอีกครั้งจนปรากฏข้อความที่ชัดเจน ดังรูปที่





รูปที่ 5 ก่อนปรับความเข้ม LCD **รูปที่ 6 หลังปรับ LCD จะได้ภาพที่ชัดเจน**

หลังจากได้ภาพที่ชัดเจนแล้ว ให้ทดลองกดปุ่มอินพุทสัญญาณไปเรื่อยๆ ควรจะได้ผลตามผลการทดลองดังนี้



รูปที่ 7 แสดงผลการทดลองเมื่อกดปุ่มไปเรื่อยๆ