BECS 31421
EXPERIMENT 05

DATE 25/4/2024

Interfacing a 16×2 LCD with a PIC Microcontroller

STUDENT NAME: G.P.D. THAMARA
STUDENT NUMBER: EC/2021/005

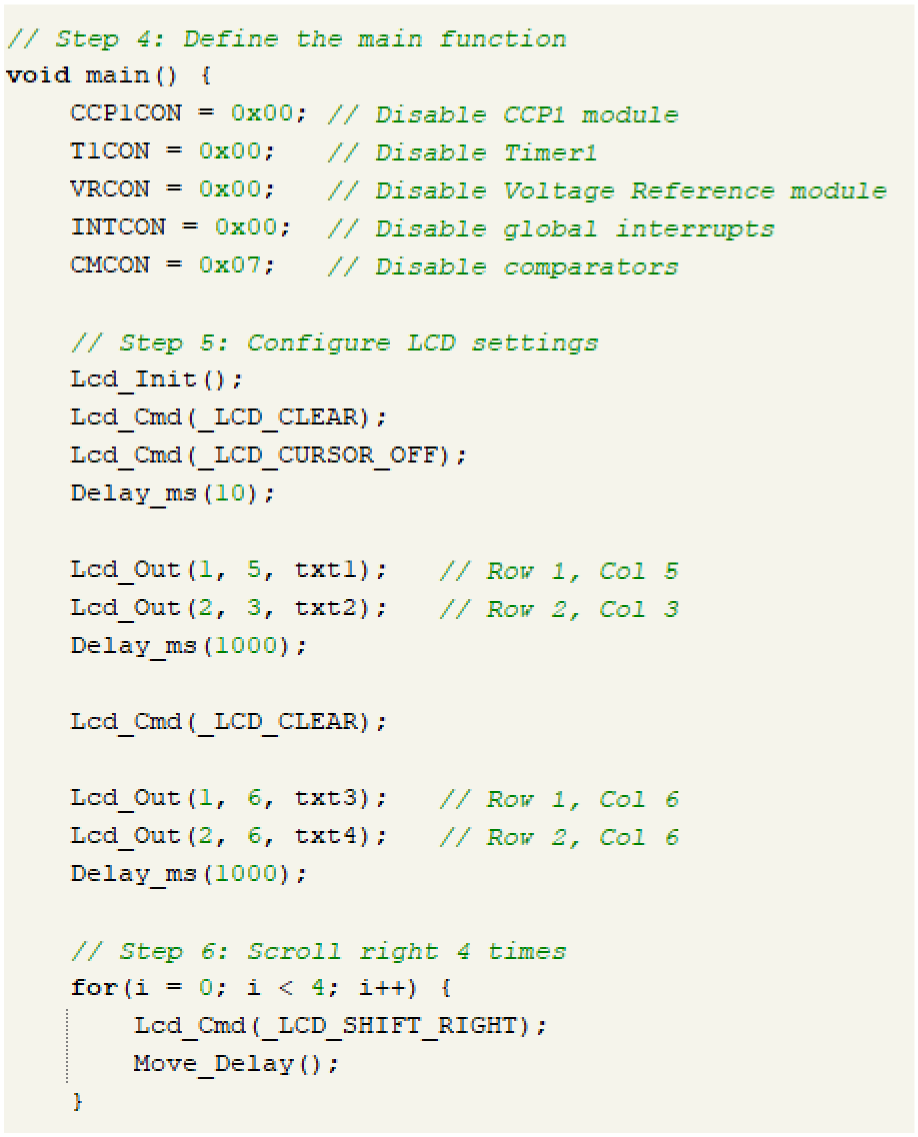
Source Code

```
// Step 1: Include necessary libraries for LCD
#include <built_in.h>

// LCD module connections
sbit LCD_RS at RB0_bit;
sbit LCD_EN at RB1_bit;
sbit LCD_D4 at RB4_bit;
sbit LCD_D5 at RB5_bit;
sbit LCD_D6 at RB6_bit;
sbit LCD_D7 at RB7_bit;

sbit LCD_RS_Direction at TRISB0_bit;
sbit LCD_EN_Direction at TRISB1_bit;
sbit LCD_D4_Direction at TRISB4_bit;
sbit LCD_D5_Direction at TRISB5_bit;
sbit LCD_D6_Direction at TRISB6_bit;
sbit LCD_D7_Direction at TRISB7_bit;

// Step 2: Define text to be displayed on the LCD
char txt1[] = "WELCOME";        // Text #1
char txt2[] = "BECS 31421";     // Text #2
char txt3[] = "EX:LCD";         // Text #3
char txt4[] = "NO:06";          // Text #4
char i; // Loop variable

// Step 3: Define a function to add delay for text moving
void Move_Delay() {
    Delay_ms(500); // 500 ms delay for scrolling effect
}

// Step 4: Define the main function
void main() {
    CCP1CON = 0x00; // Disable CCP1 module
    T1CON = 0x00;   // Disable Timer1
    VRCON = 0x00;   // Disable Voltage Reference module
    INTCON = 0x00;  // Disable global interrupts
    CMCON = 0x07;   // Disable comparators

    // Step 5: Configure LCD settings
    Lcd_Init();
    Lcd_Cmd(_LCD_CLEAR);
    Lcd_Cmd(_LCD_CURSOR_OFF);
    Delay_ms(10);

    Lcd_Out(1, 5, txt1);   // Row 1, Col 5
    Lcd_Out(2, 3, txt2);   // Row 2, Col 3
    Delay_ms(1000);

    Lcd_Cmd(_LCD_CLEAR);

    Lcd_Out(1, 6, txt3);   // Row 1, Col 6
    Lcd_Out(2, 6, txt4);   // Row 2, Col 6
    Delay_ms(1000);

    // Step 6: Scroll right 4 times
    for(i = 0; i < 4; i++) {
        Lcd_Cmd(_LCD_SHIFT_RIGHT);
        Move_Delay();
    }

    // Step 9: Scroll left 9 times
    for(i = 0; i < 9; i++) {
        Lcd_Cmd(_LCD_SHIFT_LEFT);
        Move_Delay();
    }

    // Step 10: Center text by scrolling right 5 times
    for(i = 0; i < 5; i++) {
        Lcd_Cmd(_LCD_SHIFT_RIGHT);
        Move_Delay();
    }

    // Step 11: Endless loop
    while(1) {
        // keep displaying
    }
}
```

BECS 31421 DATE 25/4/2024

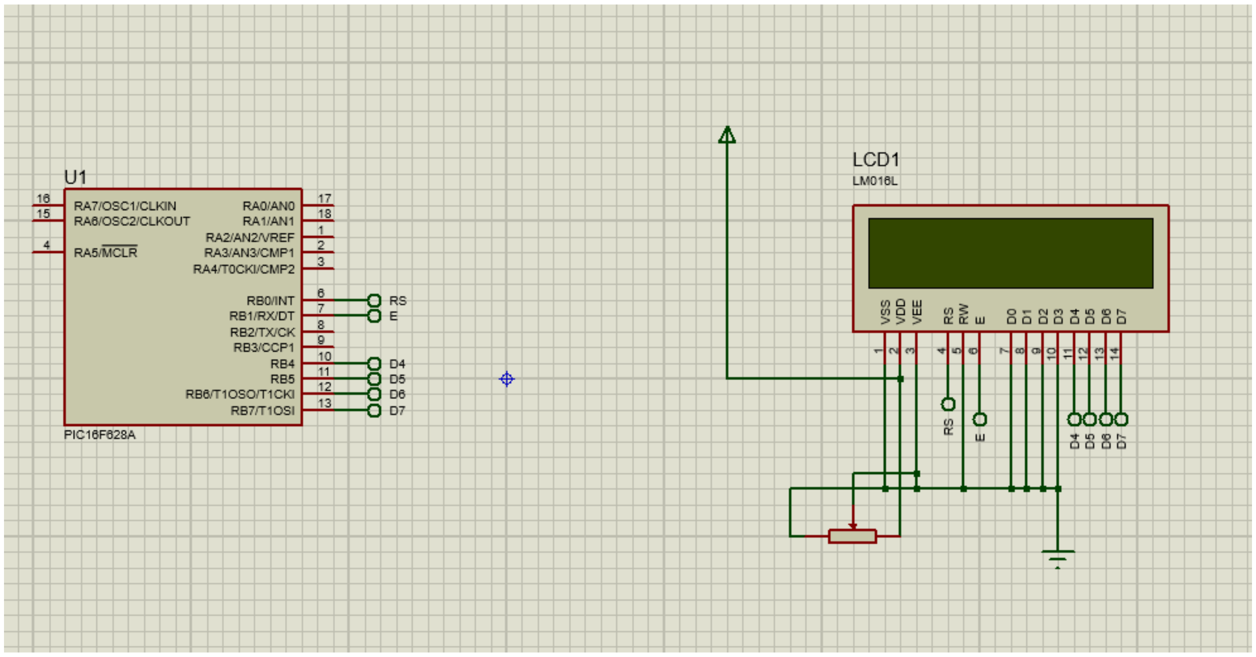
EXPERIMENT 05

```
// Step 1: Include necessary libraries for LCD
#include <built_in.h>

// LCD module connections
sbit LCD_RS at RB0_bit;
sbit LCD_EN at RB1_bit;
sbit LCD_D4 at RB4_bit;
sbit LCD_D5 at RB5_bit;
sbit LCD_D6 at RB6_bit;
sbit LCD_D7 at RB7_bit;

sbit LCD_RS_Direction at TRISB0_bit;
sbit LCD_EN_Direction at TRISB1_bit;
sbit LCD_D4_Direction at TRISB4_bit;
sbit LCD_D5_Direction at TRISB5_bit;
sbit LCD_D6_Direction at TRISB6_bit;
sbit LCD_D7_Direction at TRISB7_bit;

// Step 2: Define text to be displayed on the LCD
char txt1[] = "WELCOME";        // Text #1
char txt2[] = "BECS 31421";     // Text #2
char txt3[] = "EX:LCD";         // Text #3
char txt4[] = "NO:06";          // Text #4
char i; // Loop variable

// Step 3: Define a function to add delay for text moving
void Move_Delay() {
    Delay_ms(500); // 500 ms delay for scrolling effect
}
```



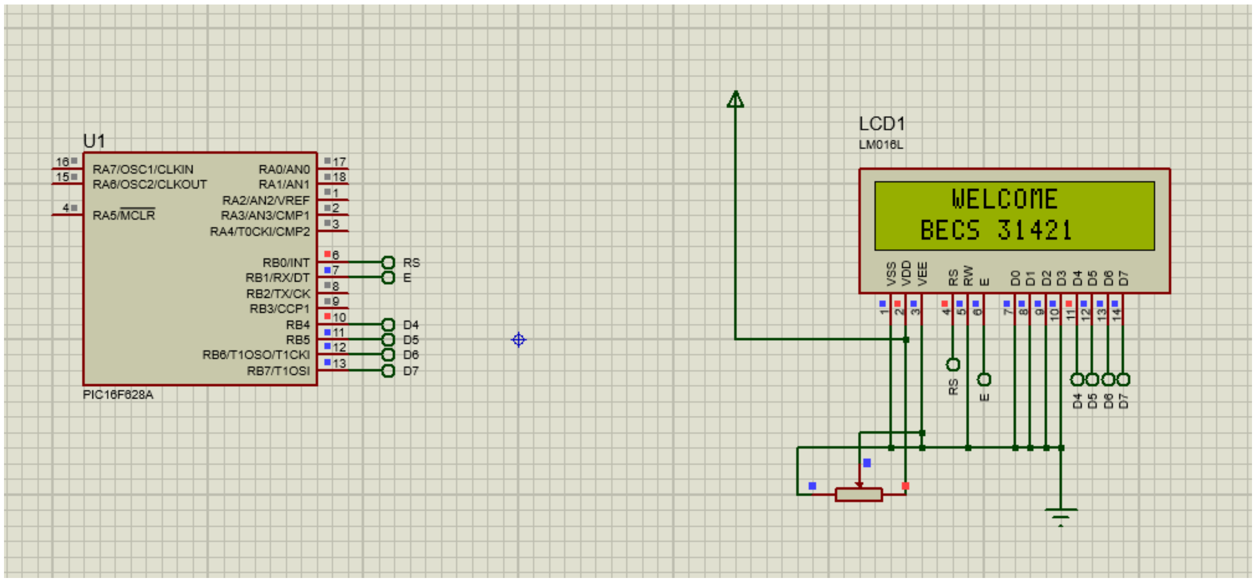
```
// Step 4: Define the main function
void main() {
    CCP1CON = 0x00; // Disable CCP1 module
    T1CON = 0x00;   // Disable Timer1
    VRCON = 0x00;   // Disable Voltage Reference module
    INTCON = 0x00;  // Disable global interrupts
    CMCON = 0x07;   // Disable comparators

    // Step 5: Configure LCD settings
    Lcd_Init();
    Lcd_Cmd(_LCD_CLEAR);
    Lcd_Cmd(_LCD_CURSOR_OFF);
    Delay_ms(10);

    Lcd_Out(1, 5, txt1);   // Row 1, Col 5
    Lcd_Out(2, 3, txt2);   // Row 2, Col 3
    Delay_ms(1000);

    Lcd_Cmd(_LCD_CLEAR);

    Lcd_Out(1, 6, txt3);   // Row 1, Col 6
    Lcd_Out(2, 6, txt4);   // Row 2, Col 6
    Delay_ms(1000);

    // Step 6: Scroll right 4 times
    for(i = 0; i < 4; i++) {
        Lcd_Cmd(_LCD_SHIFT_RIGHT);
        Move_Delay();
    }
    // Step 9: Scroll left 9 times
    for(i = 0; i < 9; i++) {
        Lcd_Cmd(_LCD_SHIFT_LEFT);
        Move_Delay();
    }

    // Step 10: Center text by scrolling right 5 times
    for(i = 0; i < 5; i++) {
        Lcd_Cmd(_LCD_SHIFT_RIGHT);
        Move_Delay();
    }

    // Step 11: Endless loop
    while(1) {
        // keep displaying
    }
}
```

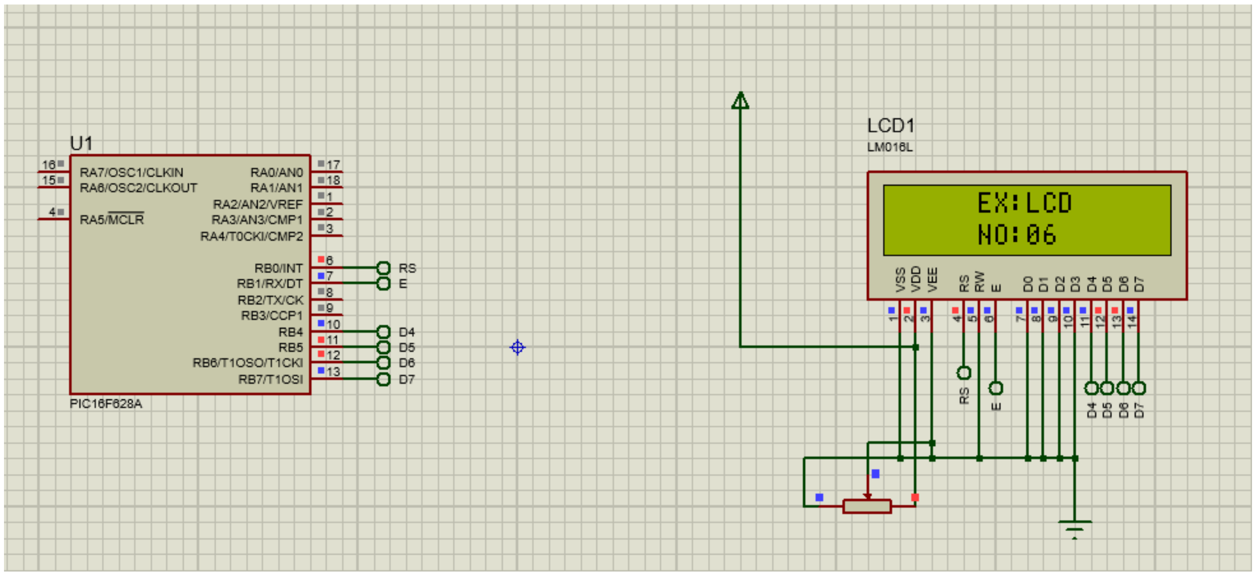
Circuit



BECS 31421 DATE 25/4/2024
EXPERIMENT 05

Observations





Discussion

A 16×2 Liquid Crystal Display functions as part of this experiment which involves connection to a PIC microcontroller for displaying static and dynamic text through programming and hardware-level pin configurations.

The main goal of this exercise is to learn practical LCD alphanumeric control techniques by writing code and setting up connections at the hardware pin level since this represents a core element in embedded systems development. The 16×2 LCD presented in this experiment possesses two horizontal lines extending across sixteen vertical positions to produce a screen capacity of thirty-two characters. The device functions with a 5V supply and sends data in 4-bit mode through DB4-DB7 data lines by moving byte information in two steps that process high-order bits first then low-order bits. Reduction of I/O pins on the microcontroller stands as the critical advantage for selecting this communication protocol when working with limited pin resources.

The LCD features three control pins which function as RS (Register Select) to alternate between command and data modes as well as RW (Read/Write) pin grounded to enable write-only mode and E (Enable) to send data to the LCD when the signal edge falls. The LCD system was programmed to show different messages as both non-moving text blocks and scrolling text during implementation. Initialization proceeded by disabling unused peripherals (CCP1 and Timer1 and Voltage Reference and Comparators and interrupts) for preventing service interference. The programmed delay periods along with LCD command sequences allowed content shifting between left and right positions to produce animation. The experiment showed students how to use low-level commands for standard LCD control and display control while they learned standard LCD command practice and experienced first-hand how correct pin mapping works with timing control in embedded system peripheral operations.