LCD Display Driver

Laboratory Report Practice 7

Sebastian Medina Garza A01139531

Ricardo Cárdenas García A01232465

Eduardo Daniel Pérez Caro A00823082

Prof. Matías Vázquez

Monterrey NL México 03/05/2022

Objectives

The main objective of this practice is to write a general-purpose 16x2 LCD display driver for the PIC18 µC and use it, along with the previously-developed 4x4 keypad driver, to implement a simple calculator on the Curiosity development board.

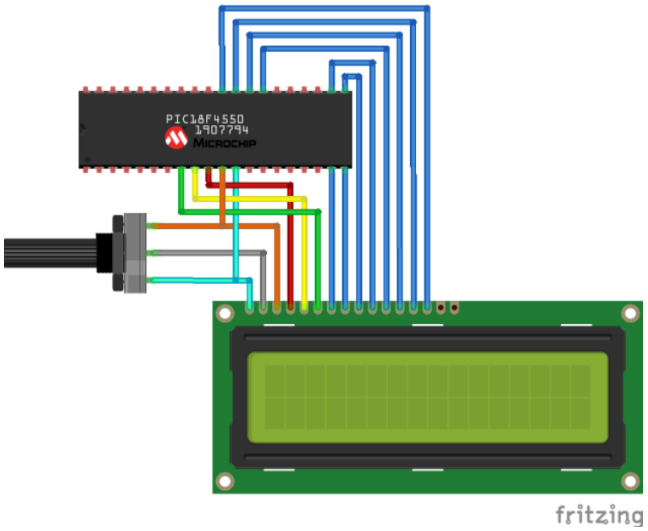
Introduction

The LCD Display Driver has 2 lines and 16 characters per line, as shown in the figure bellow.

Pin number	Symbol	Level	I/O	Function
1	V_{SS}	-	-	Power supply (GND)
2	V_{CC}	-	-	Power supply (+5V)
3	V_{EE}	-	-	Contrast adjustment
4	RS	0/1	I	0 = Instruction input, 1 = Data input
5	R/W	0/1	I	0 = Write to LCD, 1 = Read from LCD
6	E	1, 1→0	I	Enable
7	D0	0/1	I/O	Databus line 0 (LSB)
8	D1	0/1	I/O	Databus line 1
9	D2	0/1	I/O	Databus line 2
10	D3	0/1	I/O	Databus line 3
11	D4	0/1	I/O	Databus line 4
12	D5	0/1	I/O	Databus line 5
13	D6	0/1	I/O	Databus line 6
14	D7	0/1	I/O	Databus line 7

Figure 1. LCD Display pinout

The following diagram shows the wiring of the LCD for powering and µC data transmission:



Procedures

Definition of LCD constants to PIC18 ports

```
#include "Config_header.h"

#define _XTAL_FREQ 1000000

#define LCD_DATA_R      PORTD
#define LCD_DATA_W      LATD
#define LCD_DATA_DIR    TRISD
#define LCD_RS          LATCbits.LATC2
#define LCD_RS_DIR      TRISCbits.TRISC2
#define LCD_RW          LATCbits.LATC1
#define LCD_RW_DIR      TRISCbits.TRISC1
#define LCD_E           LATCbits.LATC0
#define LCD_E_DIR       TRISCbits.TRISC0
```

The following instruction illustrates how the macros are used to define the output levels for the port bits. For example, to set the RW line high (1); LCD_RS = 1;

Handling the E control line

The E line is used to tell the LCD to execute an instruction available on the data bus. It must be set high before an instruction and set low after the instruction. The E line must be manipulated whether the instruction is a read/write or if is text/instruction.

E high instruction LCD_E = 1;

E low instruction LCD_E = 0;

Checking the busy status on LCD

The Get LCD Status command determines whether the LCD is still busy executing the last instruction received. This command returns a flag through DB7: high level if the LCD is busy on current instruction execution and a low level if it is no longer busy and ready to receive and execute a new command.

```
void LCD_rdy(void){ // waits until the LCD is not busy
    char test;
    // configure LCD data bus for input
    LCD_DATA_DIR = 0b11111111;
    test = 0x80;
    while(test){
        LCD_RS = 0; // select IR register
        LCD_RW = 1; // set READ mode
        LCD_E = 1; // setup to clock data
        test = LCD_DATA_R;
        Nop();
        LCD_E = 0; // complete a READ cycle
        test &= 0x80; // check flag BUSY FLAG
    }
```

LCD initialization

The LCD must be initialized and configured before displaying data. This is accomplished by sending a few initialization instructions to the LCD. The first instruction indicates to the LCD the type of communication: and 8-bit or 4-bit data bus. The first instruction must also set the 5x8 dot character font. These two options are selected by sending the command 38h to the LCD. The RS line must be low to a command to the LCD. The following instructions send the first command. The second initialization byte is the instruction 0Fh. This instruction turns the LCD ON and the cursor ON. It is necessary to repeat part of the sequence described in the code below:

```
void LCD_init(void){
    LATC = 0;               // Make sure LCD control port is low
    LCD_E_DIR = 0;          // Set Enable as output
    LCD_RS_DIR = 0;         // Set RS as output
    LCD_RW_DIR = 0;         // Set R/W as output
    LCD_cmd(0x38);          // Display to 2x40
    LCD_cmd(0x0F);          // Display on, cursor on and blinking
    LCD_cmd(0x01);          // Clear display and move cursor home
}
```

```
void LCD_cmd(char cx){ // sends command to LCD
    LCD_rdy(); // wait until LCD is ready
    LCD_RS = 0; // select IR register
    LCD_RW = 0; // set write mode
    LCD_E = 1; // setup clock data
    Nop();
    LCD_DATA_W = cx; // send out command
    Nop(); // small delay
    LCD_E = 0; // complete an external write cycle
}
```

Main Program

The cases are defined for each key, which is stored in the numero1 then it checks if a second number has been introduced, if the operation is >0, it means a second number has been entered.

```
void main(void) {
    Ports_configuration();
    LCD_init();                                     // Se inicia el LCD
    while(1){                                       // Loop infinito
        LATB = 0x70;                                // Empezamos el MUX
        switch(PORTB){
            case 0x77:                              // MUX de la primera fila
                if(boton == 0){                     // es el primer numero de fila entonces el valor
                    numero1 = 1;                    // = 1
                    send2LCD(numero1+'0');          // Imprimimos en display
                    __delay_ms(50);                 // delay
                    boton = 1;                      //Registro de numero
                } else if (boton == 1 & operacion > 0){      //Si ya se introdujo un numero y no hay operando
                    numero2 = 1;                    // Registramos que se introdujo un segundo numero
                    send2LCD(numero2+'0');          // Imprimimos
                    __delay_ms(50);                 // delay
                }
                while (PORTB == 0);
            break;
```

Number 2, will be:

```
    case 0x7B:
        if(boton == 0){
            numero1 = 2;
            send2LCD(numero1+'0');
            __delay_ms(50);
            boton = 1;
        } else if(boton == 1 & operacion > 0){
            numero2 = 2;
            send2LCD(numero2+'0');
            __delay_ms(50);
        }
        while (PORTB == 0);
    break;
```

Sum

```
case 0x7E:
    if(boton == 1){
        operacion = 1;
        send2LCD('+');
        __delay_ms(50);
    }
    while (PORTB == 0);
break;
```

Substract

```
    case 0xBE:
        if(boton == 1){
            operacion = 2;
            send2LCD('-');
            __delay_ms(50);
        }
        while (PORTB == 0);
    break;
}
```

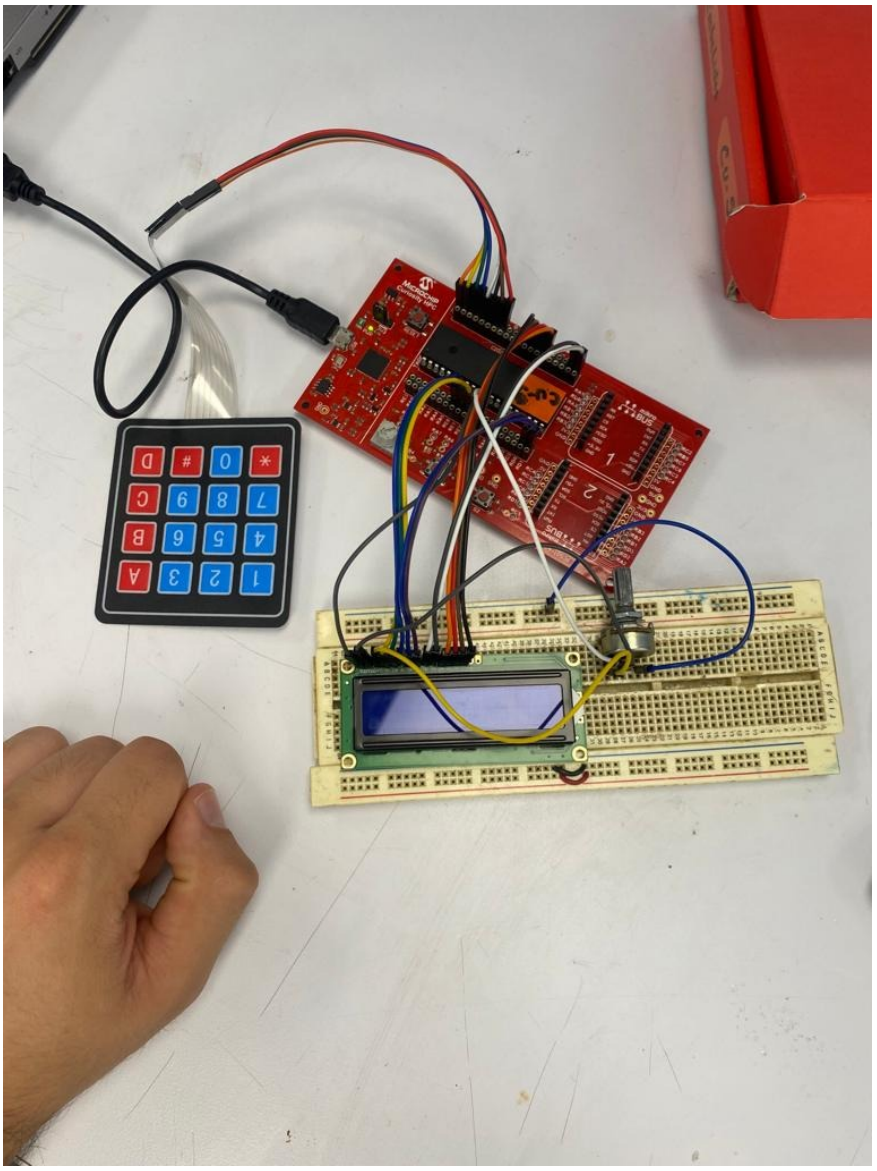
Multiplication

```
    case 0xDE:
        if(boton == 1){
            operacion = 3;
            send2LCD('*');
            __delay_ms(50);
        }
        while (PORTB == 0);
    break;
}
```

Different cases of operations

```
LATB = 0xE0;
switch(PORTB){
    case 0xE7:
        switch(operacion){
            case 1:
                res = numero1 + numero2;
            break;
            case 2:
                res = numero1 - numero2;
            break;
            case 3:
                res = numero1 * numero2;
            break;
            case 4:
                res = numero1 / numero2;
            break;
        }
        LCD_cmd(0xC1);
        send2LCD(res+'0');
        while (PORTB == 0);
    break;
```

Results



Individual conclusions

Sebastian

- In conclusion, using an LCD display is a very interesting and intuitive way to interact with the user and display numerical information.

Ricardo

- In this practice the main objective was to develop a program using the 4x4 matrix keypad driver for the PIC18 microcontroller, the usage of 7 segment display was a very helpful and educating way to see theory apply in real life.

Eduardo

- For this practice we used the 4x4 matrix keypad driver in order to develop a program that allows us to enter a random number so the program has to store it, and also has the ability to receive another one.

References

- Matiaz, V. (2022). Introduction to MPLAB X IDE. Retrieved from: **https://github.com/matias-vazquez/microcontroladores/tree/main/Lab06**