Instituto Tecnológico y de Estudios Superiores de Occidente – ITESO

ITESO

Universidad Jesuita de Guadalajara

Materia: Fundamentos de Microprocesadores y Microcontroladores

Profesor: Álvaro Gutiérrez Arce

Fecha: 03/05/2022

Tema: Práctica 3: LCD y Teclado Matricial

Carrera: Sistemas computacionales

Autor: José Jaime Gutiérrez Martínez

Oscar Javier Hernández Salamanca

Contenido

Desarrollo teórico

Una pantalla LCD, a diferencia de los LED, puede mostrar números y todos los caracteres ASCII junto con algunos caracteres especiales. Esto significa que son adecuados para mostrar cosas como relojes o calendarios donde necesita mostrar números o letras en posiciones específicas en la pantalla. En esta práctica la interfaz entre la pantalla y el microcontrolador es simple ya que todos los caracteres se monitorean automáticamente en los circuitos de control interno de los módulos LCD.

Desarrollo

Implementar un programa que permita desplegar en el LCD la información que se introduzca a través del teclado. Se deben filtrar los rebotes del teclado para evitar múltiples lecturas de la misma tecla y evitar que el display parpadeé.

Cada vez que se introduzca un nuevo carácter debe colocarse a la derecha del anterior. Una vez que se llena la primera línea debe saltar al inicio de la segunda. Cuando la segunda línea se llene, debe borrar la pantalla y regresar al inicio de la primera.

Además, se debe agregar un botón que genere una interrupción que permita introducir caracteres tipo ASCII. Al dejar presionado el botón, se podrán introducir dos dígitos en hexadecimal a través del teclado y se mostrara el carácter ASCII correspondiente en el display LCD. Por ejemplo, si no se presiona este botón y se oprime la tecla 4 enseguida se oprime la tecla 5, en el display LCD se mostrará un 4 y a la derecha un 5. Ahora bien, al presionar este botón, y oprimir la tecla 4 enseguida se oprime la tecla 5, en el display LCD se mostrará el carácter E; en código ASCII el carácter E tiene un código igual a 45 en hexadecimal.

Finalmente se debe agregar un segundo botón que genere una interrupción para que la información que se está desplegando en la pantalla, se transmita vía el puerto serial del micro a una computadora personal conectada al sistema por medio de un módulo de bluetooth inalámbrico.

Esquemático completo

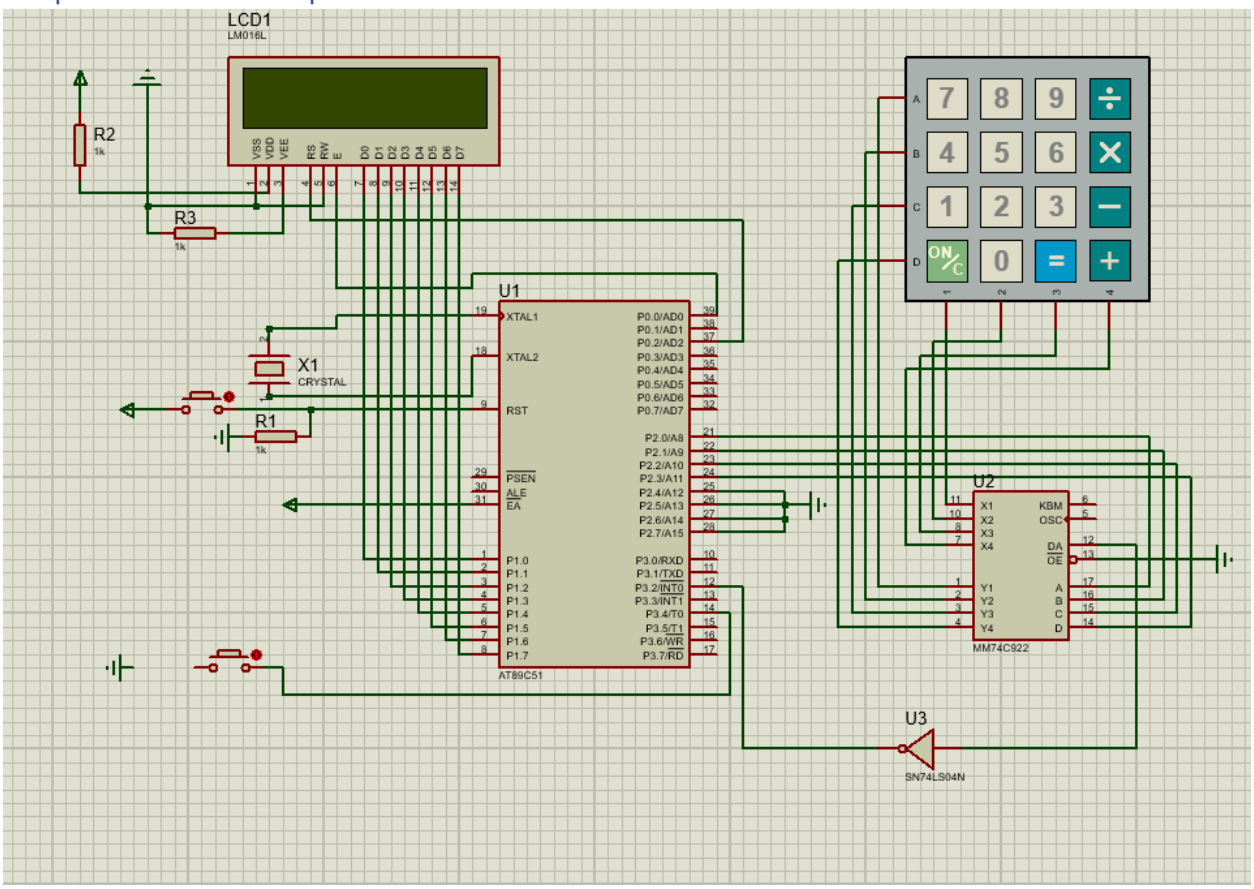

Diagrama de flujo

Código del programa con comentarios

```
LCD_E               EQU P0.0
LCD_RW              EQU P0.1
LCD_RS              EQU P0.2
LCD_DISPLAY_BUS     EQU P1
KEYBOARD_INPUT      EQU P2
DELAY_SHORT_VAR     EQU R2
DELAY_LONG_VAR      EQU R3
LCD_VAR             EQU R4
RAM_POINTER         EQU R0
TEMP_RAM_POINTER    EQU R1
ASCII_REGISTER      EQU R5
KEYBOARD_COUNTER    EQU R6
ALT_PIN             EQU P3.4

ORG 0000H
JMP MAIN
//interrupciones
ORG 0003H
    JMP ISR_1
//main
ORG 00040H
MAIN:
    ACALL RESET_MEMORY
    ACALL INIT_LCD_DISPLAY
    ACALL INIT_INTERRUPTIONS

MAIN_1:
    JNB ALT_PIN, $
    INC KEYBOARD_COUNTER
    JB ALT_PIN, $
    CJNE KEYBOARD_COUNTER, #00H, $
    JMP MAIN_1

//Interrupcion de teclado
ISR_1:
        //si KEYBOARD_COUNTER == 2 entonces estamos ALT, segundo caracter en LCD
        CJNE KEYBOARD_COUNTER, #02H, isr_01
        MOV A, LCD_VAR
        ADD A, KEYBOARD_INPUT
        MOV LCD_VAR, A
        MOV KEYBOARD_COUNTER, #00H
        //mandamos datos al lcd
        ACALL LCD_SEND_DATA
    RETI

// si es el primer digito, KEYBOARD_COUNTER ==1 entonces ponermos en acumulador para despues sumer
    isr_01:
        CJNE KEYBOARD_COUNTER, #01H, isr_normal
        MOV A, KEYBOARD_INPUT
        SWAP A
        MOV LCD_VAR, A
        //KEYBOARD_COUNTER a 2 para ir al paso de arriba la siguiente ves
        INC KEYBOARD_COUNTER
    RETI

    //tomamos input, guardamos en acumulador y lo enviamos
    isr_normal:
        MOV A, KEYBOARD_INPUT
        ACALL CHANGE_TO_ASCII
        MOV LCD_VAR, A
        ACALL LCD_SEND_DATA

    RETI

//reseteamos toda la memoria
RESET_MEMORY:
    MOV LCD_VAR, #00H
    MOV DELAY_SHORT_VAR, #00H
    MOV DELAY_LONG_VAR, #01H
    //RAM
    MOV RAM_POINTER, #030H
    MOV KEYBOARD_COUNTER, #00H
    SETB ALT_PIN
    CLR LCD_RS
    CLR LCD_RW
    CLR LCD_E
    CLR TI


    MOV TEMP_RAM_POINTER, #30H

    memory_reset_loop:
        MOV @TEMP_RAM_POINTER, #00H
        INC TEMP_RAM_POINTER
        CJNE TEMP_RAM_POINTER, #50H, memory_reset_loop

    RET

RESET_RAM_AND_LCD:
    MOV TEMP_RAM_POINTER, #30H
    MOV LCD_VAR, #01H
    ACALL LCD_SEND_INSTRUCTION

    ACALL LCD_CHANGE_ROW_TO_1

    reset_ram_and_lcd_loop:
        MOV @TEMP_RAM_POINTER, #00H
        INC TEMP_RAM_POINTER
        CJNE TEMP_RAM_POINTER, #50H, reset_ram_and_lcd_loop


    RET
```

```
//Inicializamos LCD
INIT_LCD_DISPLAY:
    //pausas para inicializar
    MOV LCD_VAR, #38H
    ACALL LCD_SEND_INSTRUCTION
    ACALL DELAY_LONG

    MOV LCD_VAR, #38H
    ACALL LCD_SEND_INSTRUCTION
    ACALL DELAY_LONG

    MOV LCD_VAR, #38H
    ACALL LCD_SEND_INSTRUCTION
    ACALL DELAY_LONG

    //prendemos display
    MOV LCD_VAR, #01H
    ACALL LCD_SEND_INSTRUCTION
    ACALL DELAY_LONG

    //cursor en modo parpadeo
    MOV LCD_VAR, #0FH
    ACALL LCD_SEND_INSTRUCTION
    ACALL DELAY_LONG
    RET

//enviamos a la pantalla dato
LCD_SEND_DATA:
    SETB LCD_RS
    MOV LCD_DISPLAY_BUS, LCD_VAR
    SETB LCD_E
    NOP //Con este NOP de buffer aseguramos que se tarde los 500ns necesarios.
    CLR LCD_E

    //Hacemos una pausa corta para darle tiempo a la pantalla de procesar.
    ACALL DELAY_SHORT
    //Guardamos el dato enviado en la RAM para cuando tengamos que enviarlo por serial.
    ACALL STORE_LCD_VAR

    RET

//Envia lo que esté en LCD_VAR como una instrucción.
LCD_SEND_INSTRUCTION:
    CLR LCD_RS //RS debe estar en bajo para que la pantalla interprete lo recibido como dato.
    MOV LCD_DISPLAY_BUS, LCD_VAR
    SETB LCD_E
    NOP //Con este NOP de buffer aseguramos que se tarde los 500ns necesarios.
    CLR LCD_E
    RET

//Habilita los vectores de interrupción.
INIT_INTERRUPTIONS:
    MOV IE, #10000101B
    SETB IT0
    SETB IT1
    RET

//Guarda lo que esté en LCD_VAR en la ubicación en donde apunta RAM_POINTER y lo avanza una ubicación.
STORE_LCD_VAR:
    MOV A, LCD_VAR
    MOV @RAM_POINTER, A
    INC RAM_POINTER

    //Si nuestro RAM_POINTER llegó a 40H, ya escribimos 16 datos y necesitamos cambiar el display a la
segunda fila.
    CJNE RAM_POINTER, #40H, NO_CHANGE_1
    ACALL LCD_CHANGE_ROW_TO_2

    NO_CHANGE_1:
        //Si nuestro RAM_POINTER llegó a 50H, llegó al final de la segunda fila porque ya escribimos 32
datos. Hay que reiniciarlo y mover el display a
        //la primera fila.
        CJNE RAM_POINTER, #50H, NO_CHANGE_2
        ACALL RESET_RAM_AND_LCD
        MOV RAM_POINTER, #30H
        ACALL RESET_RAM_AND_LCD

    NO_CHANGE_2:
        RET


//Hace que el apuntador de la pantalla LCD vaya a la segunda fila.
LCD_CHANGE_ROW_TO_2:
    MOV LCD_VAR, #0C0H
    ACALL LCD_SEND_INSTRUCTION
    RET

//Hace que el apuntador de la pantalla LCD vaya a la primera fila.
LCD_CHANGE_ROW_TO_1:
    MOV LCD_VAR, #80H
    ACALL LCD_SEND_INSTRUCTION
    RET


//Envia lo que está en el acumulador por el puerto serial y espera a que termine.
SEND_ACC_SERIAL:
    MOV SBUF, A
    JNB TI, $
    CLR TI
    RET

//Transforma un input de 4 bytes a un caracter ASCII.
CHANGE_TO_ASCII:
    MOV ASCII_REGISTER, A
    ascii_step_0:
        CJNE A, #00H, ascii_step_1
        MOV A, ASCII_REGISTER
        ADD A, #30H
        RET
    ascii_step_1:
        CJNE A, #0AH, ascii_step_2
        MOV A, ASCII_REGISTER
        ADD A, #37H
        RET
    ascii_step_2:
        DEC A
        SJMP ascii_step_0

//long delay
DELAY_LONG:
    MOV DELAY_LONG_VAR, #15H

//short delay
DELAY_SHORT:
    MOV DELAY_SHORT_VAR, #0FFH
    DJNZ DELAY_SHORT_VAR, $
    DJNZ DELAY_LONG_VAR, DELAY_SHORT
    MOV DELAY_LONG_VAR, #1D //Le ponemos 1 para que si hacemos un Delay Short en el futuro, no vaya a
decrementar a DelayLongVar a FF
    RET

END
```

Conclusiones individuales

José Jaime Gutiérrez Martínez

Esta práctica me ayudó a comprender cómo utilizar la comunicación serial y darle una aplicación más real al microcontrolador al permitirme ver los datos entregados en la computadora. Tuvimos demasiados problemas con el cableado, cambiando múltiples componentes obteniendo finalmente el correcto funcionamiento de la práctica, aunque solamente nos faltó implementar el módulo bluetooth, se logró el objetivo de la práctica.

Oscar Javier Hernández Salamanca

Esta práctica me ayudó a comprender cómo transmitir datos a la pantalla LCD, cómo utilizar un teclado matricial, cómo pasar del teclado matricial a la pantalla LCD, así como también el realizar interrupciones. La pantalla LCD y el teclado matriciál son esenciales para esta práctica ya que puedes examinar lo que está ocurriendo y si estás escribiendo lo que quieres que se escriba porque tienes una respuesta concreta de que es correcto o incorrecto. Me ayudó a completar mi comprensión de varios conceptos cubiertos en clase, así como un uso tangible de un microcontrolador.

Referencias

Manish K Patel (2014). The 8051 Microcontroller Based Embedded Systems. McGraw Hill Education