Práctica 7
Display LCD

Islas Bravo, Andrés.
Estudiante - Laboratorio de Microcontroladores.
Departamento de Arquitectura e Ingeniería
Instituto Tecnológico y de Estudios Superiores de Monterrey
Ciudad de México, México
a01339391@itesm.mx

Valverde López, Juan Pablo.
Estudiante - Laboratorio de Microcontroladores.
Departamento de Arquitectura e Ingeniería
Instituto Tecnológico y de Estudios Superiores de Monterrey
Ciudad de México, México
a01656127@itesm.mx

Zhu Chen, Alfredo.
Estudiante - Laboratorio de Microcontroladores.
Departamento de Arquitectura e Ingeniería
Instituto Tecnológico y de Estudios Superiores de Monterrey
Ciudad de México, México
a01651980@itesm.mx

Abstract

This practice covers a lot of concepts with the main objective of controlling an LCD display. During this practice the approach was oriented into developing a right structure for an LCD controller, this includes the research to take a decision between two modes of sending information (4 bit or 8 bit). Also the consideration of Times that the LCD requires in order to do the tasks that are ordered.

I. Nomenclatura

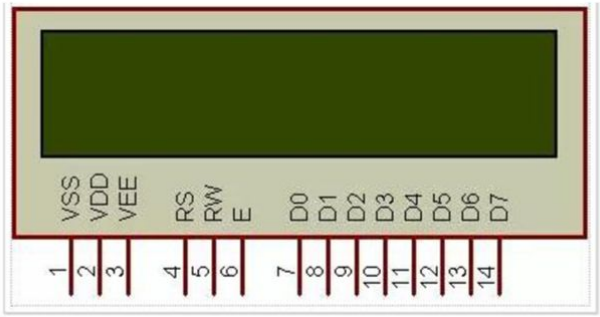
Unidad de microcontrolador,	*MCU*.
Resistencia-Capacitor,	*RC*.
Unidad de frecuencia, hercio,	*Hz*.
Diodo Emisor de Luz,	*LED*.

II. Introducción

El presente documento aborda todo lo acontecido durante el control de un LCD. Se muestra todo lo que se necesita para controlar en un modo de 4 bits el display. Esto va desde su configuración inicial donde se le indica que estará trabajando en dicho modo y posteriormente se mencionan todas las instrucciones necesarias para completar su configuración, esto sin olvidar que dicha configuración será mandada en bloques de 4 bits, correspondientes a la parte alta y baja de lo que normalmente sería una instrucción de 8 bits. Asimismo las líneas de códigos presentes corresponden a un mensaje de "Hola Mundo!", esto con una distribución donde el "Hola" se encuentra en la primera fila y el "mundo" en la segunda, ambas centradas. El motivo de este arreglo es meramente para un mayor dominio de las memorias que están presentes en el LCD, y el manejo de una de una mayor cantidad de instrucciones para obtener una mayor familiarización con el instrumento.

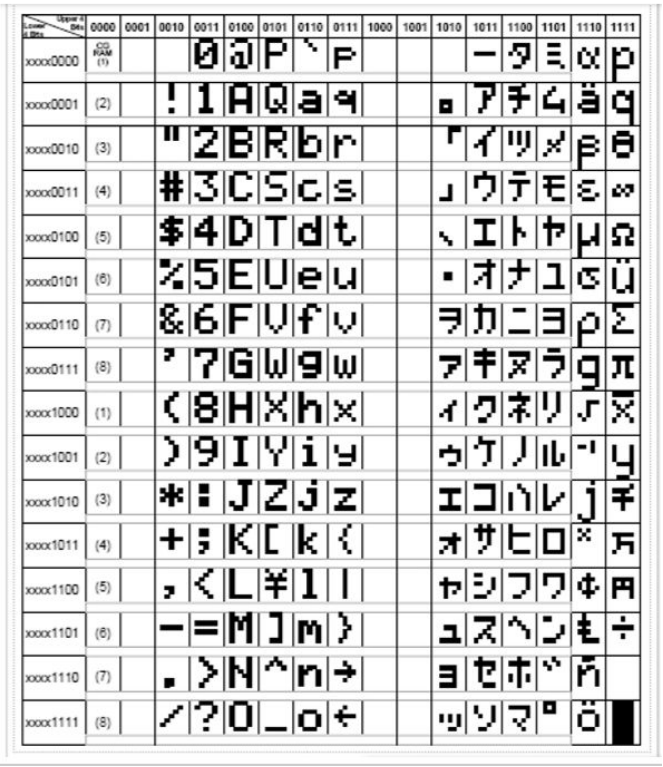
III. Marco Teórico

Los displays LCD (Liquid Crystal Display) más comunes que se pueden encontrar son de una, dos o cuatro líneas y soportan alrededor de 80 caracteres debido a su controlador interno. Los que tienen un controlador consisten en 14 pines de las cuales 3 son de alimentación (V_{SS},V_{DD},V_{EE}), 3 de control(RS,RW,E) y 8 de datos (D_7, D_6, D_5, D_4, D_3, D_2, D_1, D_0). Tal como se muestra en la siguiente captura, de izquierda a derecha los pines corresponden a: tierra de alimentación, alimentación de 5V, ajuste del contraste, bit para indicar si se envían datos o instrucciones, bit de escritura o lectura, habilitación de la señal y 8 bits de la información a enviar[1].



Captura 1: LCD de 14 pines. Tomado desde:https://www.8051projects.net/lcd-interfacing/introduction.php

Los LCDs pueden escribir caracteres de acuerdo con una tabla CGROM (Generador de Caracteres en ROM) que básicamente es para saber qué se puede escribir a la pantalla de acuerdo con un código en byte. Para ello se utiliza la siguiente tabla de código ASCII para saber lo que se mostrará en el LCD[1].



Captura 2: Tabla ASCII para LCD. Tomado desde:https://www.8051projects.net/lcd-interfacing/basics.php

Otra de las consideraciones importantes en el momento de trabajar con LCDs, es la configuración propia para que el controlador interno sepa cómo operar en su modo que se desee. En la práctica, se realiza una inicialización para que todo esté preparado en su funcionamiento, este puede incluir: función para modo de trabajo, control de la pantalla y limpiar la pantalla[1].

IV. Desarrollo

La presente práctica propone el uso de una pantalla LCD de 2 líneas con capacidad de escritura de 16 características, para la visualización de un mensaje de bienvenida. Este tipo de pantallas son de las más básicas y cuentan con un microcontrolador integrado para su funcionamiento. El que supone un reto es la sincronización entre los tiempos que tarda un microcontrolador en ejecutar ciertas tareas frente al tiempo del ATMega16. Para llevar a cabo esta práctica se tiene dos subrutinas principales para que se pueda visualizar un mensaje en la pantalla LCD, una es la de inicialización y otra la de envío de los caracterectreres a la pantalla. Es importante señalar que el método de trabajo de esta pantalla será el de 4 bits, es importante hacer esta aclaración pues también cuenta con un modo de trabajo de 8 bits sin embargo, al tratarse del ATMega16, un problema recurrente será la administración del número de puertos para poder llevar a cabo la resolución de problemas propuestos. A partir de ahora, a la subrutina de envío se le llamará "sendLCD_4bits", la que será encargada de:

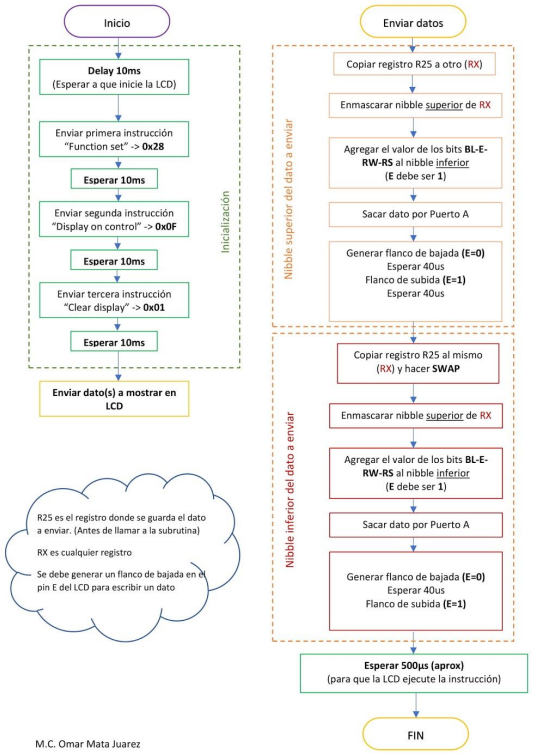
- Obtener un registro Rx, siendo cualquier número del 16 al 26, el dato de 8 bits a enviar a la LCD. Usar el concepto de definiciones para poder modificar el registro Rx por cualquier otro (".def Rx = datos"
- La bandera T debe indicar si se enviará un dato o una instrucción.
    - T = 0 (Instrucción)
    - T = 1 (Dato)
- Adecuar el dato de 8 bits para que la LCD lo pueda recibir en modo 4 bits.

Para la rutina de inicialización, a la que a partir de ahora se la referirá como "sendLCD_4bits", se debe:

- Configurar la LCD en modo 4 bits, usar 2 líneas y Font 5x7.
- Habilitar el modo incremental del cursor y apagar el desplazamiento "shift"
- Encender el display, el cursor y poner el cursor en modo "blink"

Es importante mencionar que esta subrutina hará uso de la anterior subrutina descrita.

Para un mejor entendimiento, se tomó los diagramas de flujo de las dos subrutinas:



Captura 1: Diagrama de Flujo Subrutinas. Autoría: M.C. Omar Mata Juárez.

Es necesario preparar la pila que hará uso de las últimas direcciones de la memoria RAM. Para este programa sí será de utilidad la pila ya que cada vez se vuelven más críticos la administración de recursos, en este caso específico de registros. En programas anteriores se utilizaban en promedio más de 4 registros para poder emplear los retardos por software, ahora solo son necesarios 2, empleando las instrucciones de PUSH/POP.

```
7    ;pila
8    ldi r16, low(ramend)
9    out spl, r16
10   ldi r16, high(ramend)
11   out sph, r16
```

Captura 1: Configuración Pila.

```
delay10ms:
    push r20
    push r21
    LDI R20,104
    ciclo2:
        LDI R21,255
    ciclo1:
        DEC R21
        BRNE ciclo1
        DEC R20
        BRNE ciclo2
    pop r21
    pop r20
RET
```

```
delay40us:
    push r20
    push r21
    LDI R20,24
    ciclo3:
        LDI R21,3
    ciclo4:
        DEC R21
        BRNE ciclo1
        DEC R20
        BRNE ciclo2
    pop r21
    pop r20
RET
```

Captura 2. Uso de la Pila

Una vez se emplea el diagrama de flujo de la Captura 1, el problema se vuelve sencillo, además de entender cómo funciona en su interior el display y los modos de trabajo que el microcontrolador con el que viene ofrece. Como se mencionó con anterioridad, para poder emplear cualquier registro en una futura ocasión, se hizo uso del ".def".

```
.def dato = r25
```

Captura 3: Definición del Registro r25.

La preparación de los puertos es el siguiente paso cuando se programa un microcontrolador. Una vez completado este paso fundamental, se inicia la descripción del programa.

```
13   call initlcd_4bits
14   set
15   ldi dato,'*'
16   call sendlcd_4bits
17   call espacios
18   ldi dato,'H'
19   call sendlcd_4bits
20   ldi dato,'o'
21   call sendlcd_4bits
22   ldi dato,'l'
23   call sendlcd_4bits
24   ldi dato,'a'
25   call sendlcd_4bits
26   call espacios
27   ldi dato,'*'
28   call sendlcd_4bits
```

Captura 4.1: Código - Main. Inicialización - Escritura de Caracteres

```
28   call sendlcd_4bits
29   clt
30   ldi dato,0b11000000;salta de línea
31   call sendlcd_4bits
32   set
33   ldi dato,'*';
34   call sendlcd_4bits
35   call espacios2
36   ldi dato,'M'
37   call sendlcd_4bits
38   ldi dato,'u'
39   call sendlcd_4bits
40   ldi dato,'n'
41   call sendlcd_4bits
42   ldi dato,'d'
43   call sendlcd_4bits
```

Captura 4.2: Código - Escritura de Caracteres.

```
44   ldi dato,'o'
45   call sendlcd_4bits
46   ldi dato,'!'
47   call sendlcd_4bits
48   call espacios2
49   ldi dato,'*'
50   call sendlcd_4bits
51   fin: rjmp fin
52   espacios2:
53       ldi r19, 4
54       bucle2:
55       set
56       ldi dato,' '
57       call sendlcd_4bits
58       dec r19
```

Captura 4.3: Código - Escritura de Caracteres

```
59       brne bucle2
60   ret
61   espacios:
62       ldi r19, 5
63       bucle:
64       set
65       ldi dato,' ';ascii espacio
66       call sendlcd_4bits
67       dec r19
68       brne bucle
69   ret
```

Captura 4.4: Código - Escritura de Caracteres - sendlcd_4bits.

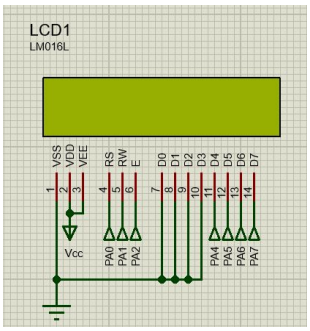
Para comprobar el funcionamiento del sistema, se utilizará Proteus en su versión 8.8, donde se carga el archivo .hex que es generado en el directorio del proyecto.

Comprobación por Proteus.

Para esta comprobación se hará uso del programa anterior diseñado, el sistema mínimo del ATMega16, que básicamente consiste en el etiquetado y programación de un botón de RESET para el MCU.

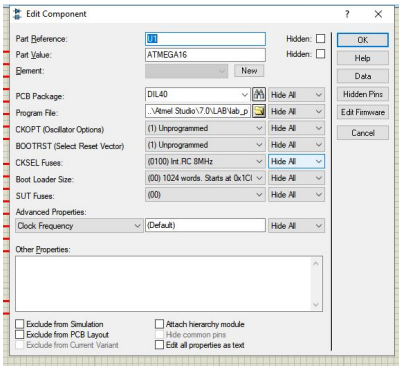
Las salidas para ese sistema será el módulo LM016L, que integra la pantalla LCD con su microcontrolador.

La conexión del módulo es:



Captura 5: Módulo LM016L que integra pantalla LCD con su microcontrolador..

Para cargar el archivo de instrucción de código máquina al MCU, se da click derecho sobre él y se selecciona en "Edit Properties". Donde emergerá una ventana, allí en "Program File" se carga la dirección del programa con extensión .hex de donde se servirá el MCU para simular el comportamiento.

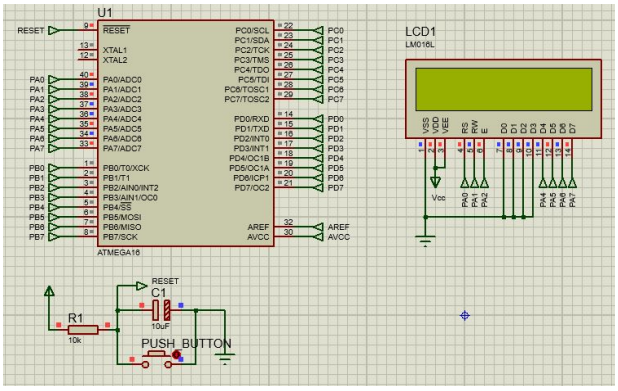


Captura 6: Carga de archivo ".hex" para simulación.

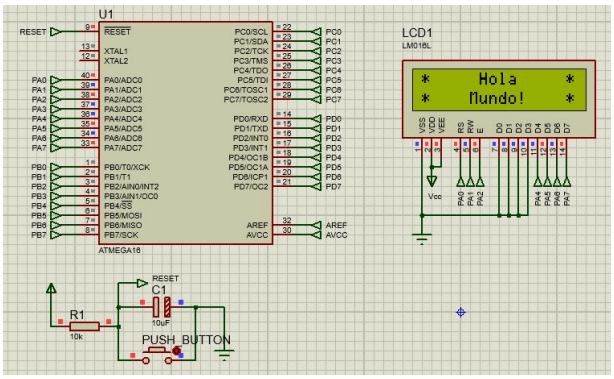
Para la selección del la frecuencia a la que trabajará el MCU, se lo hace en el apartado de "CKSEL Fuses" que prepara al MCU a trabajar de cierta forma, en este caso específico "0100", lo que significa que funcionará con su reloj interior, hecho con un arreglo de RC a 8MHz.

V. Resultados

A. *Switch 1 - Corrimiento Secuencia A*



Captura 7: Cargado e Iniciación de Simulación.



Captura 8: Visualización del mensaje en la pantalla LCD.

VI. Conclusiones

Andrés Islas Bravo:

Esta práctica permitió retomar conceptos que se habían manejado en previas materias de semestres anteriores, pero en esta ocasión abordarlos desde la perspectiva de un lenguaje ensamblador sobre un ATMega 16. Dentro de lo más importante y nuevo respecto conocimientos fue la parte de enviar en modo serial las instrucciones, con el propósito de sólo usar 4 bits. Esto fue importante ya que se tuvieron que considerar nuevos parámetros de tiempos entre cada instrucción y entender a mayor profundidad la comunicación entre el controlador y el display.

Juan Pablo Valverde López:

Entender el funcionamiento y la interacción entre dos dispositivos puede ser complicada sin los conocimientos previos necesarios. Para esta práctica una de las cosas más complicada fue la sincronización entre ambos MCUs para que interactúen y se pueda ver el mensaje en la pantalla. Es interesante cómo el criterio de diseño y administración de recursos cada vez son aspectos más importantes para tomar en cuenta en cómo se aborda el problema. Fuera de las pequeños retrasos con respecto a las rutinas de inicialización y envío de caracteres, el problema se abordó con éxito.

Alfredo Zhu Chen:

Esta práctica me permitió mostrar un mensaje en un LCD. Fue muy interesante ya que el LCD es un periférico de salida que tiene mucha utilidad y tiene su forma especial para trabajar con ella. Una de las complicaciones que tuve al programar con este dispositivo fueron los retardos, ya que el mismo LCD cuenta con un controlador que opera con requisitos de retardos específicos para que funcione Pude reforzar y aprender conocimientos con respecto a su manejo. Finalmente, logré probar el código escrito en el programa de simulación de Proteus para observar el funcionamiento de manera visual.

VII. Referencias

[1] S. a. (2014). LCD Interfacing Tutorial. Obtenido desde: https://www.8051projects.net/lcd-interfacing/introduction.php

[2] ATMEL. (2010). 8-bit AVR Microcontroller with 16K Bytes In-System Programmable Flash – AtMega Atmega16L. [Archivo PDF]. Obtenido desde: http://ww1.microchip.com/downloads/en/devicedoc/doc2466.pdf