Universidad Nacional de San Luis
Facultad de Ciencias Físico Matemáticas y Naturales
Ingeniería Electrónica con orientación en Sistemas Digitales
Técnico Universitario en Microprocesadores
Profesorado en Tecnología Electrónica

LABORATORIO DE INTERFACES

PRÁCTICO Nº 3

Interface con Teclados y Displays

Índice:

TRABAJO PRÁCTICO Nº 3

TECLADOS - DISPLAYS

1. Objetivos

- **Estudiar el funcionamiento de Teclados independientes y Matriciales.**
- **Realizar la conexión de teclados a un microcontrolador.**
- **Programar Módulos LCD Inteligente 16x2.**

2. Material de Referencia

- **Apuntes de la Cátedra.**
- **"Microcontroller Projects in C for the 8051" - Dogan Ibrahim**
- **Hojas de datos.**

3. Listado de Materiales

1 Microcontrolador AT89C2051	**1 Cristal 12MHz**
1 Circuito integrado 74LS08	**1 Módulo LCD Inteligente 16x2**
1 Preset 5K	**2 cap 33pF**
1 R 8K2	**1 cap 10uF x 25V**

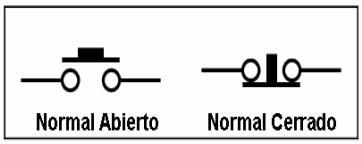
4. Listado de Instrumental

- **1 Tester digital**
- **1 Entrenador LAB - MC**
- **1 Bornera de Puerto Paralelo**
- **1 Programador Universal ChipMax**

5. Conceptos básicos

La mayoría de los sistemas basados en microprocesador o microcontrolador incluyen dispositivos de entrada como interruptores, teclados, llaves rotativas de varias posiciones, etc. y dispositivos de salida como leds, displays 7 segmentos, LCD, LCD programable, etc. para comunicarse con el usuario del sistema.
Estos dispositivos le permiten al usuario intercambiar información con el sistema, ingresando datos por medio de un teclado y observando resultados a través de un display.
Los interruptores son dispositivos en los cuales mediante un determinado mecanismo son capaces de abrir o cerrar un circuito eléctrico generando así una señal para su posterior tratamiento. Consisten en una o varias láminas que mediante un movimiento mecánico abren o cierran un circuito y generan un nivel lógico.

Los interruptores vienen en dos variantes: NA (Normal Abierto) y NC (Normal Cerrado). El interruptor NA al pulsarse se cierra, y el NC al pulsarse se abre.
Los interruptores electromecánicos, por su misma naturaleza, nunca realizan conmutaciones limpias. De hecho, cuando un interruptor se abre o se cierra, sus superficies de contacto oscilan varias veces entre un estado y el otro antes de unirse o separarse definitivamente. Este fenómeno se denomina rebote y puede causar serios problemas cuando se utilizan interruptores como dispositivos de entrada de sistemas digitales sensibles a cambios de nivel como flip-flops, contadores, micros, etc.



Pulsador

Se denomina pulsador al dispositivo mecánico que acciona un interruptor. En muchas ocasiones los términos pulsador o tecla se usan en forma indistinta.

Teclados

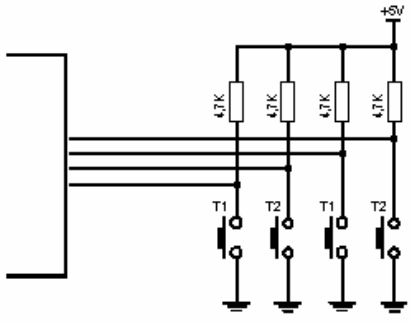
El teclado es un dispositivo de entrada formado por varios pulsadores. Cada pulsador representa un carácter, número, o función. Los teclados pueden tener tres configuraciones básicas: teclados independientes, teclados matriciales y teclados mixtos

Teclados Independientes

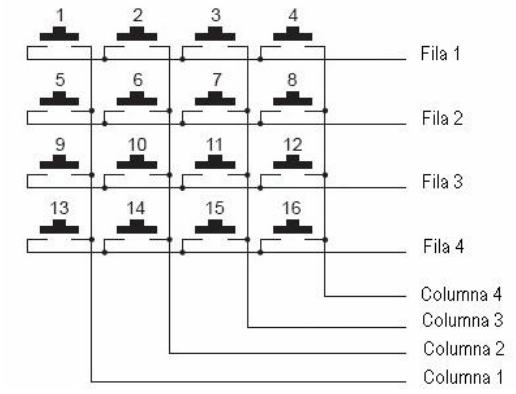
Los teclados independientes están formados por varios interruptores que poseen un extremo común disponible externamente, y un terminal externo por cada interruptor.
La figura siguiente muestra como se puede conectar directamente un teclado independiente a una puerta de entrada de un microcontrolador.

Teclado Matricial

El teclado matricial está formado por teclas dispuestas en una matriz de filas y columnas conductoras. Un extremo de cada interruptor se conecta a una columna, y el otro, a una fila. El teclado dispone externamente de un número de conductores dado por la suma de las filas y columnas del teclado.



Teclado independiente.



Teclado matricial.

Exploración

Las filas se conectan a una puerta de salida y las columnas a una puerta de entrada. Por la puerta de salida se sacan todos 1 lógicos a excepción de una línea por la cual se saca un 0 lógico. Este 0 lógico se desplaza de una línea a otra.
Cada vez que se saca una secuencia por el puerto de salida, se lee el puerto de entrada. Si todas las entradas están en 1 lógico significa que no se presionó ninguna tecla. Si hay una línea de entrada que está en 0, significa que se presionó una tecla. La tecla presionada está ubicada en la fila por la cual se sacó un 0 lógico, y la columna por la cual se leyó un 0 lógico.

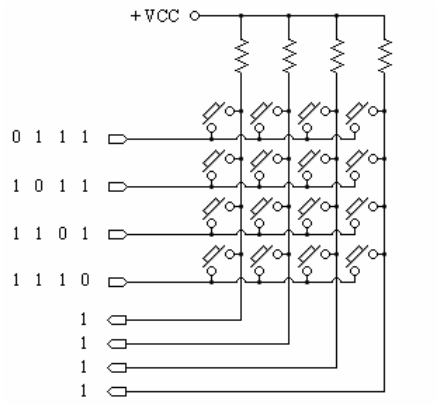
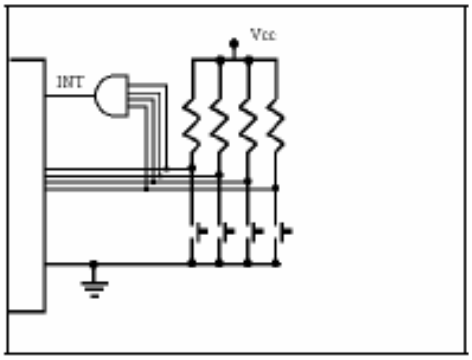


Fig. 4. Exploración de un Teclado Matricial.

Atención de un teclado

Un teclado se puede atender por consulta (pooling) realizando la lectura de su estado en forma permanente o periódica, o bien por medio de interrupciones.
Cuando se emplea pooling el microprocesador distrae tiempo que puede emplear para realizar otras tareas.
La mayor parte de las veces en que el microprocesador consulta el estado del teclado, éste no tendrá ninguna tecla presionada. Es decir que atiende al teclado aún cuando este no necesita atención. Cuando se emplea una interrupción para atender el teclado, éste genera una señal de validación para indicar que se ha presionado una tecla, y en la rutina de interrupción se efectúa la lectura del estado del teclado para determinar que tecla se presionó. Cuando se utiliza interrupción, el microprocesador puede realizar otras tareas y atender el teclado solo cuando este lo requiera.
La figura siguiente muestra como se conecta un teclado independiente a un sistema basado en microprocesador por medio de interrupciones.



Eliminación de rebotes de los contactos

Cuando se abre o cierra un interruptor se producen rebotes, lo que produce una secuencia de pulsos transitorios en la entrada de un dispositivo lógico. Si el interruptor o teclado se conecta a una entrada de interrupción se producirán varias interrupciones, y se lo utiliza para incrementar un contador este producirá un salto irregular en la cuenta.

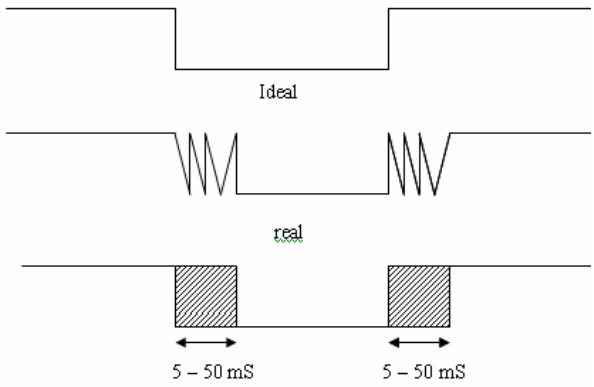
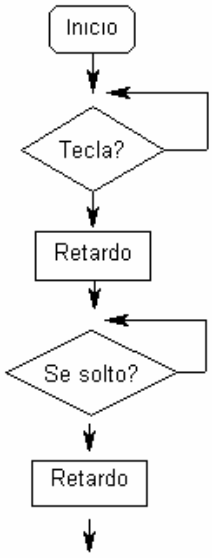


Fig. 6. Rebote en los contactos.
Para eliminar estas transiciones producidas por los rebotes de los contactos se pueden emplear varios métodos:

- **Empleo de un dispositivo con entrada Schmitt Trigger.**
- **Empleo de flip flop R-S.**
- **Empleo de circuitos integrados específicos (MAX6818, MC14490).**
- **Empleo de demoras por software (usando timers).**

Eliminación de rebotes por software

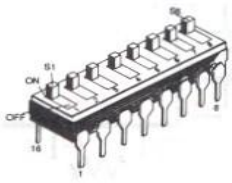
Podemos enmascarar el rebote por software. Nuestro programa debe detectar el cierre del interruptor y esperar hasta que finalice el tiempo de rebotes antes de aceptar el estado definitivo. Una vez cerrado debemos hacer lo mismo cuando se abre el interruptor ya que otra vez existen rebotes.
El tiempo de rebotes varía dependiente del tipo de contactos, la presión ejercida sobre el interruptor, etc., pero en la práctica varia entre 5 y 50mS.
El siguiente ejemplo espera que se presione una tecla. Cuando esta abierta entrega un "1", cuando se presiona un "0".




```
while (Tecla == 1);   // Espera presión de tecla
 retardo();           // Elimina rebotes
 while (Tecla == 0);  // Espera que se suelte
  retardo();          // Elimina rebotes

Tecla esta definido como : sbit Tecla = P3^7;
```

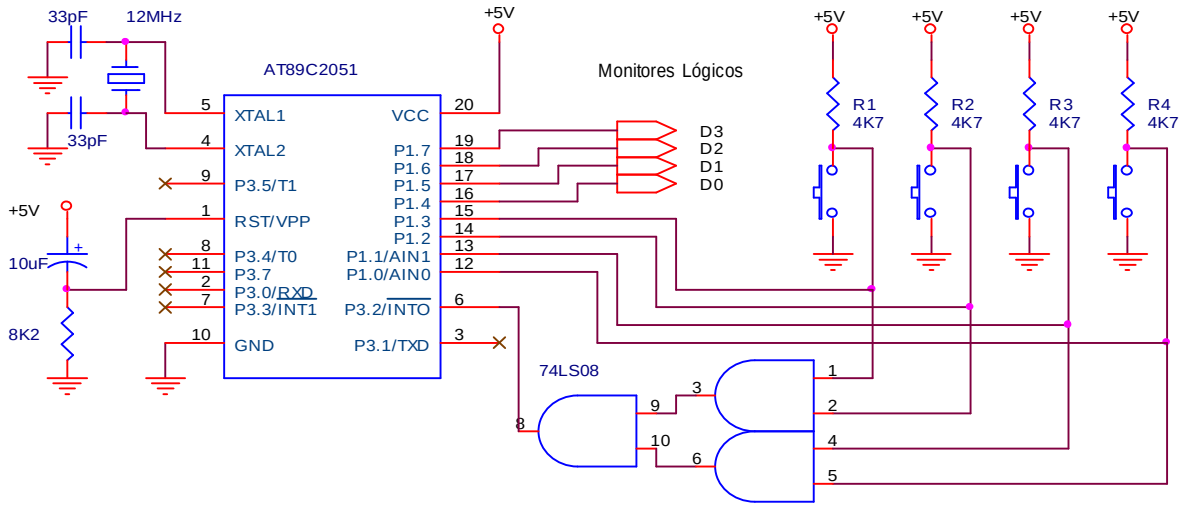
6. Desarrollo de la práctica.

6.1 Se le proveerán distintos interruptores. Identifique los contactos de los mismos con el tester y realice un esquema que lo represente.

Denominación	Aspecto	Esquema
Micro-Switch **Son muy útiles como fines de carrera. Vienen en distintos tamaños y para todas las potencias.**		
Llaves **Son las más comunes. Se usan para encendido de equipos. Poseen uno o más inversores.**		
Tact Switch **Existe una gran variedad de tamaño y forma. Se usan en TV, Audio, etc.**		
DIP-Switch **Se utilizan para realizar seteos como por ejemplo en placas de Adquisición para setear la dirección base.**		
Llave selectora **Permiten seleccionar entre varias salidas posibles. Pueden tener mas de 1 polo y hasta 12 posiciones.**		

6.2 Conexión de teclado independiente a un microcontrolador

6.2.1 Arme el siguiente circuito en el entrenador LAB-MC.
6.2.2 Simule el programa EJ01.C en el uVision2. Verifique la interrupción en el puerto3.
6.2.3. Grabe el programa EJ01.hex en el microcontrolador y verifique funcionamiento.

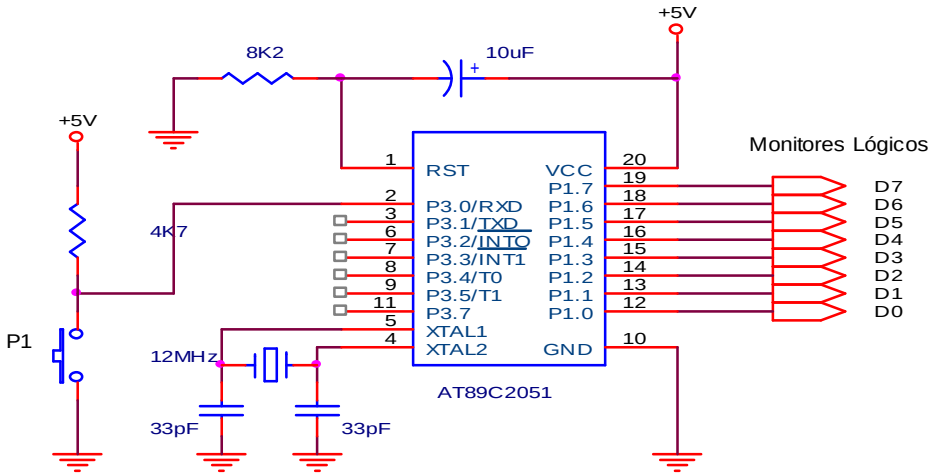



```
#include <AT892051.h>

#define ETX0  0
unsigned char Dato;

// Rutina de Atención de Interrupción Externa 0

void Tecla (void) interrupt ETX0{
  IE   = 0x00;                // Inhabilita /INT0
   Dato = ((P1)&0x0f);        // Leo estado interruptores
   Dato = Dato * 16;          // Paso a la parte alta (cambio nibbles)
   P1   = ((P1&0x0f)|Dato);   //
  IE   = 0x81;                // Habilita /INT0
 } // ------------------------------------------------------------

// Programa Principal

void main (void){
 IT1 = 1;                     // Configura /INT0 por Flanco de Bajada
 IE = 0x81;                   // Habilita  /INT0
 for(;;);
}
```

6.2.4 ¿Explique como se detecta la pulsación de una tecla en este circuito. ¿Por que se utiliza una interrupción externa?

. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .

6.2.5. Arme el siguiente circuito en el entrenador LAB-MC. Conecte el Puerto 1 a los monitores lógicos, estos mostraran el valor de un contador binario.
6.2.6. Grabe el programa Rebote.hex en el microcontrolador y verifique.




```
// ********************************************************************
// * Proyecto    : Rebote de interruptores                            *
// ********************************************************************

#include <AT892051.h>

#define ETX0  0
unsigned char Dato;

// Rutina de Atención de Interrupción Externa 0

void Tecla (void) interrupt ETX0{

  IE   = 0x00;               // Inhabilita /INT0
   Conta ++;                 // Incrementa contador
   P1   = ~ (Conta);         // Lo saca por el puerto
  IE   = 0x81;               // Habilita /INT0
 } // ------------------------------------------------------------

// Programa Principal

void main (void){
 IT1 = 1;                    // Configura /INT0 por Flanco de Bajada
 IE = 0x81;                  // Habilita  /INT0
 for(;;);
}
```

6.2.7 ¿Qué sucede con el contador cuando se presiona el pulsador una única vez? Explique este fenómeno.

. . . . . .
. . . . . .
. . . . . .

6.3. Eliminar los rebotes de contactos por software

6.3.1. Utilizando el circuito anterior, grabe en memoria del micro el programa EJ4.HEX.
6.3.2. Verifique como se incrementa el contador ahora.

```
#include <AT892051.h>

#define ETX0  0
sbit Tecla = P3^2;
unsigned char Dato;

//  Rutina que genera un retardo de N x 1 mS

void Retardo (int N){
 int K;
 TMOD = 0x01;               // Timer 0 en Modo 1
 for (K=1;K<=N;K++){
   TF0=0;
    TH0 = 0x0FC;            // Recarga Timers
    TL0 = 0x17;             // para 1 mS  TH=0FC TL=17
    TR0 = 1;                // Habilita TIMER 0
   while(!TF0);             // Espera sobrepasamiento
   }
  TR0=0;                    // Detiene Timer 0
} // ------------------------------------------------------------

// Rutina de Atención de Interrupción Externa 0

void Externa0 (void) interrupt ETX0{
  IE   = 0x00;               // Inhabilita /INT0
   Retardo (40);              // Retardo 40 mS
   Conta ++;                 // Incrementa contador
   P1   = ~ (Conta);         // Lo saca por el puerto
   while (Tecla == 0);        // Espera que se suelte
   Retardo (40);             // Retardo 40 mS
  IE   = 0x81;               // Habilita /INT0
 } // ------------------------------------------------------------

// Programa Principal

void main (void){
 IT1 = 1;                   // Configura /INT0 por Flanco de Bajada
 IE = 0x81;                 // Habilita  /INT0
 for(;;);
}
```

6.3.3 ¿Con la demora de este programa se eliminaron los rebotes o tubo que utilizar otro valor de demora?

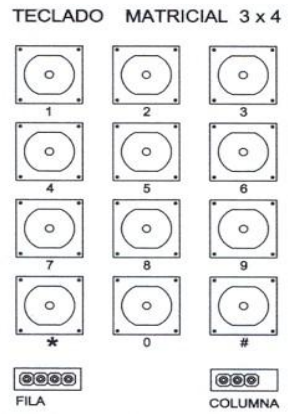
. . . . . .
. . . . . .
. . . . . .

6.3.4 Explique como se eliminan los rebotes con el programa anterior.

. . . . . .
. . . . . .
. . . . . .
. . . . . .

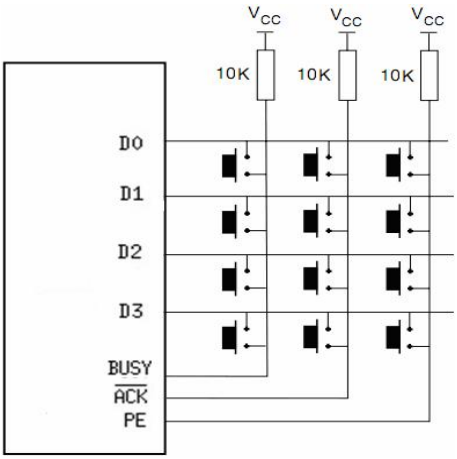
6.4 Conexión de teclado matricial al Puerto paralelo de la PC

6.4.1. Primero verificaremos el funcionamiento del teclado matricial con un tester.
6.4.2. Coloque el tester en la posición de continuidad. Ubique una punta en una de las filas y la otra en una de las columnas. Presione la tecla correspondiente a esta intersección. Realice lo mismo para varias teclas.



La siguiente aplicación nos permitirá comprender el funcionamiento de un teclado matricial y la forma de realizar la exploración del mismo.

6.4.2 Conecte el teclado matricial del LAB-MC a la bornera del puerto paralelo del siguiente modo:



6.4.3 Ejecute la aplicación "Teclado LPT" y pruebe el funcionamiento del programa.

6.4.4 Obtenga el código de exploración que se envía por los bits D0...D3.

. . . . . .
. . . . . .
. . . . . .

6.4.4 Explique por que colocaron diodos en las líneas de datos. ¿Que puede suceder si se sacan estos diodos?

. . . . . .
. . . . . .
. . . . . .

6.4.5 Obtenga el código que identifica a cada tecla:

Tecla	Código	Tecla	Código
1		7	
2		8	
3		9	
4		0	
5		*	
6		#	

6.5 Uso de Displays

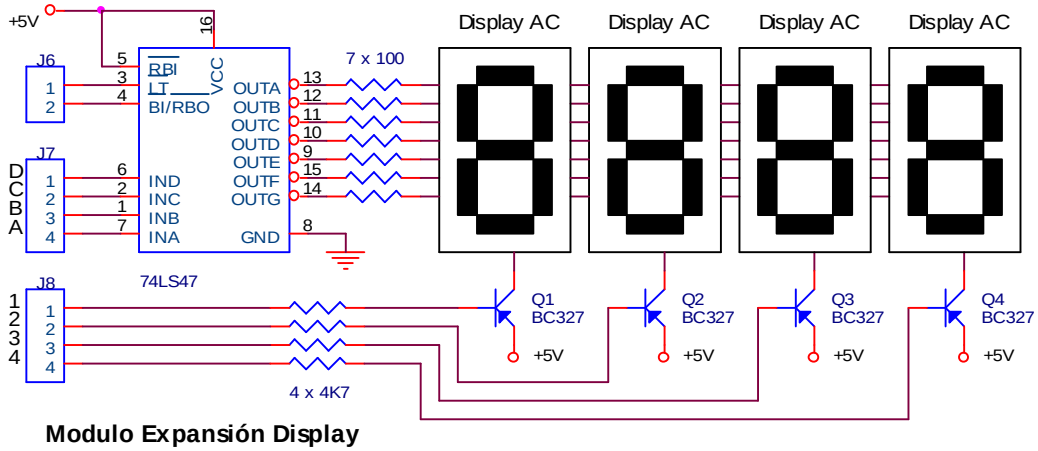
Los display son los medios por los cuales visualizamos información proveniente de un circuito digital. Existen de diversas tecnologías y prestaciones. En este laboratorio veremos los más utilizados en aplicaciones con microcontroladores.
Utilizaremos la técnica del multiplexado que nos permite ahorrar líneas y veremos algunas características de esta técnica.
Realizaremos una aplicación con display multiplexado y otra con Módulo LCD 16x2 o también llamado LCD inteligente.

6.5.1 Realizar el multiplexado de displays.

La técnica del multiplexado se basa en el hecho de que la visualización por turno de un dígito diferente por vez, realizado a una frecuencia adecuada, resulta indistinguible de la visualizacion simultánea de todos los digitos. La secuencia de multiplexado sería:

a. Inhabilitar la totalidad de los dígitos.
b. Colocar el código BCD correspondiente al digito n.
c. Habilitar ahora el digito n.
d. Repetir para los demás digitos (volver al punto (a) con n+1)

6.5.2 Utilice el Módulo Display Multiplexado del LAB-MC. El circuito consiste de un display multiplexado de 4 dígitos. El control se efectua por el Puerto Paralelo de la PC. Compile el código C y ejecute el programa. El mismo debe mostrar el dato "1234".



Modulo Expansión Display Multiplexado x 4

```
#include <AT892051.h>

sbit DIG1   = P1^4;  sbit DIG2   = P1^5;
sbit DIG3   = P1^6;  sbit DIG4   = P1^7;

unsigned char
DISP[10]={0xF0,0xF1,0xF2,0xF3,0xF4,0xF5,0xF6,0xF7,0xF8,0XF9};
unsigned char U,D,C,M,orden=1;


// Rutina de Inicialización del TIMER 0

void Timer_Init(){
  TMOD = 0x01;              // Programa TIMER 0 en Modo 1
  TH0  = 0x88;              // Carga Registros del TIMER
  TL0  = 0x30;              // TH = 88 TL = 30
  TR0  = 1;                 // Habilita TIMER 0
 } // ------------------------------------------------------------


void Mostrar(void) interrupt 1  // RSI del TIMER 0
{
  TH0 = 0x88;               // Recarga Timers
  TL0 = 0x30;
  P1  = 0xFF;               // Inhabilita todos los dígitos

 switch(orden){             // Selecciona dígito

   case 1:                  // Habilita el digito 1
    P1 = DISP[U] ;          //  ___     _________________________
    DIG1 = 0;               //     |___|
   break;

   case 2:                  // Habilita el digito 2
    P1 = DISP[D] ;          //  _________     ____________________
    DIG2 = 0;               //           |___|
   break;

   case 3:                  // Habilita el digito 3
    P1 = DISP[C] ;          //  _______________     _______________
    DIG3 = 0;               //                |___|
   break;

   case 4:                  // Habilita el digito 4
    P1 = DISP[M] ;          //  ____________________     _________
    DIG4 = 0;               //                     |___|
   break;
   }
   orden++;  if (orden==5) orden=1;
}


void main(void) {
  U = 4; D = 3; C = 2; M = 1;  // Inicializa Dígitos
   Timer_Init();               // Programa TIMER 0
   IE = 0x82;                  // Habilita Interrupción TIMER 0
   for (;;);                   // Espera Interrupción
 }  // ------------------------------------------------------
```

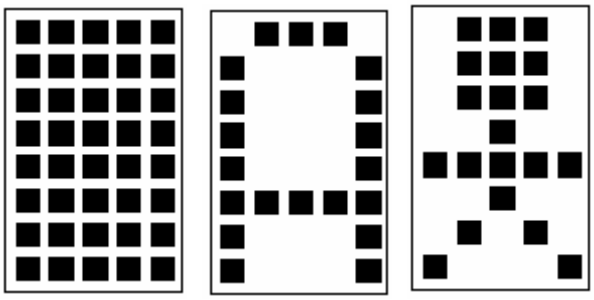
6.5.3 Aumente y disminuya los retardos. ¿Qué sucede al aumentarlos? ¿Qué sucede al disminuirlos? ¿Qué sucede con el brillo de los digitos? ¿Por qué?

. . . . . .
. . . . . .
. . . . . .
. . . . . .

6.6 Controlar un display programable 16X2.

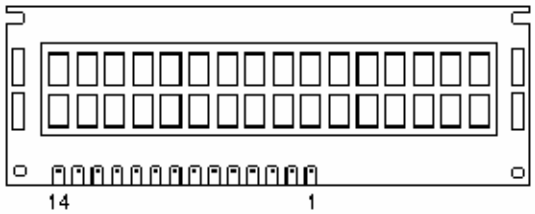
El LCD tiene un aspecto físico como el mostrado en siguiente figura. Está constituido por un circuito impreso en el que se hallan integrados el controlador de display, el display LCD, otros componentes y los pines para la conexión. Sobre el circuito impreso se encuentra el LCD en sí, rodeado por una estructura metálica que lo protege. En total se pueden visualizar 2 líneas de 16 caracteres cada una, es decir, 2x16 = 32 caracteres, como se muestra en la figura de abajo.
La tensión nominal de alimentación es de 5V, con un consumo menor de 5mA.

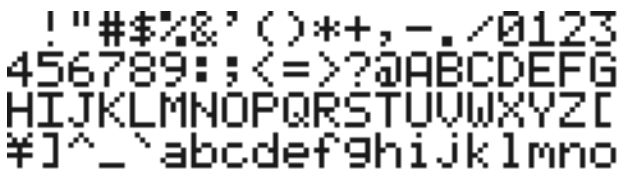
Aspecto de un Módulo LCD 16x2



Caracteres 5x8

El LCD dispone de una matriz de 5 x 8 puntos para representar cada carácter. En total se pueden representar 256 caracteres diferentes. 240 caracteres estan grabados dentro del LCD y representan las letras mayúsculas, minúsculas, signos de puntuación, números, etc... Existen 8 caracteres que pueden ser definidos por el usuario. En la anterior se muestra gráficamente cómo es la matriz de representación de los caracteres. Se ha dibujado el carácter A y un carácter definido por el usuario.

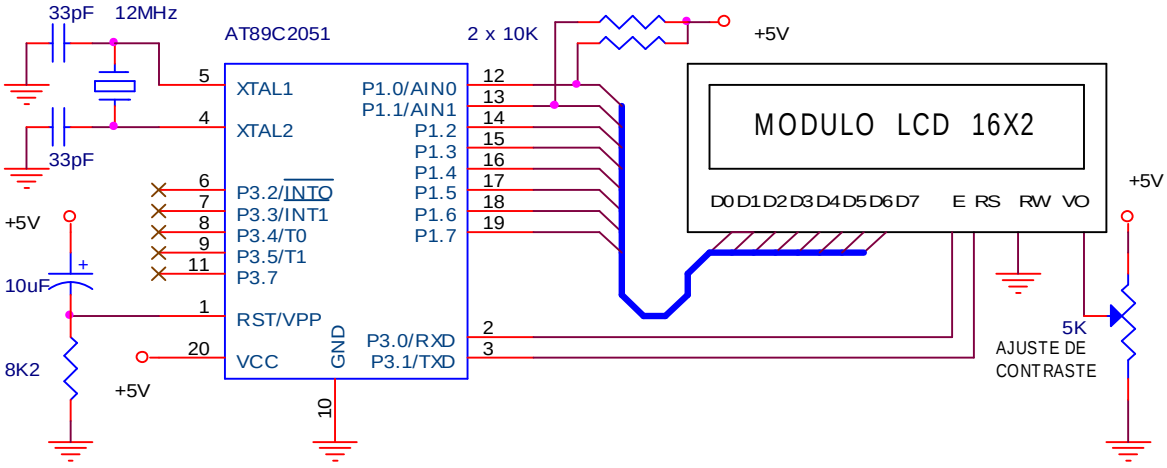




Distribución de terminales de un Módulo LCD 16x2

Nro PIN	Nombre	E/S	Descripción
1	VSS	Power	Tierra
2	VDD	Power	5 V
3	VO	Analog	Control de Contraste
4	RS	Entrada	Selección Registro
5	R/W	Entrada	Read/Write
6	E	Entrada	Enable
7	D0	E/S	Dato LSB
8	D1	E/S	Dato
9	D2	E/S	Dato
10	D3	E/S	Dato
11	D4	E/S	Dato
12	D5	E/S	Dato
13	D6	E/S	Dato
14	D7	E/S	Dato MSB

6.6.1 Arme el siguiente circuito y verifíquelo con la hoja de datos del Módulo LCD, ya que dependiendo del fabricante algunos poseen los terminales VCC y GND invertidos.



6.6.2 Grabe en la memoria del micro el programa LCD8BIT.HEX y verifique el funcionamiento del mismo. Posiblemente deba ajustar el contraste del LCD.

6.6.3 Utilizando la hojas de datos del LCD modifique el programa anterior para realizar lo siguiente:

a. Debe mostrar en la primera fila su Apellido y en la segunda su numero de registro.

b. Agregue desplazamiento automático de caracteres.

c. Presionando un pulsador debe borrar el display y mostrar la carrera que estudia.

Para aprobar la práctica debe presentar ante el docente a cargo:

1. Implementación funcionando de todos los ítems.
2. Cuestionario de Lab 03 completo (Jueves 01/10).

7. Anexo I. Resumen de comandos del LCD.

Command	Code										Description	Execution Time
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
Clear Display	0	0	0	0	0	0	0	0	0	1	Clears the display and returns the cursor to the home position (address 0).	82µs~1.64ms
Return Home	0	0	0	0	0	0	0	0	1	*	Returns the cursor to the home position (address 0). Also returns a shifted display to the home position. DD RAM contents remain unchanged.	40µs~1.64ms
Entry Mode Set	0	0	0	0	0	0	0	1	I/D	S	Sets the cursor move direction and enables/disables the display.	40µs
Display ON/OFF Control	0	0	0	0	0	0	1	D	C	B	Turns the display ON/OFF (D), or the cursor ON/OFF (C), and blink of the character at the cursor position (B).	40µs
Cursor & Display Shift	0	0	0	0	0	1	S/C	R/L	*	*	Moves the cursor and shifts the display without changing the DD RAM contents.	40µs
Function Set	0	0	0	0	1	DL	N$	F	*	#	Sets the data width (DL), the number of lines in the display (L), and the character font (F).	40µs
Set CG RAM Address	0	0	0	1	A_{CG}						Sets the CG RAM address. CG RAM data can be read or altered after making this setting.	40µs
Set DD RAM Address	0	0	1	A_{DD}							Sets the DD RAM address. Data may be written or read after making this setting.	40µs
Read Busy Flag & Address	0	1	BF	AC							Reads the BUSY flag (BF) indicating that an internal operation is being performed and reads the address counter contents.	1µs
Write Data to CG or DD RAM	1	0	Write Data								Writes data into DD RAM or CG RAM.	46µs
Read Data from CG or DD RAM	1	1	Read Data								Reads data from DD RAM or CG RAM.	46µs
	I/D = 1: Increment I/D = 0: Decrement S = 1: Accompanies display shift. S/C= 1: Display shift S/C = 0: cursor move R/L= 1: Shift to the right. R/L= 0: Shift to the left. DL = 1: 8 bits DL = 0: 4 bits N = 1: 2 lines N = 0: 1 line F = 1: 5x10 dots F = 0: 5 x 7 dots BF = 1: Busy BF = 0: Can accept data										DD RAM: Display data RAM CG RAM: Character generator RAM A_{CG}: CG RAM Address A_{DD}: DD RAM Address Corresponds to cursor address. AC: Address counter Used for both DD and CG RAM address.	Execution times are typical. If transfers are timed by software and the busy flag is not used, add 10% to the above times.

8. Anexo II. Programa ejemplo de control a 8 líneas.

```
// ***********************************************************************
// *  Programa   : Ejemplo para manejo de un modulo LCD 16X2             *
// *  Modo LCD   : Modulo a 8 bits 2 lineas sin desplazamiento           *
// *  Conexion   : D0..D7 =  P1.0..P1.7  (P1.0,P1.1  Pull-Up 4K7)        *
// *  E (Enable) : P3.0 - RS         : P3.1                              *
// ***********************************************************************


#include <AT892051.h>

void Send_Control(void);

unsigned char Dato;
int i;
char Mensaje [32]={" LABORATORIO DE INTERFACES 2007"};
sbit  E         = P3^0;
sbit  RS        = P3^1;


// Rutina que genera un retardo de N por 1 mSeg

void Demora (int N){
 int K;
 TMOD = 0x01;                  // Timer 0 en Modo 1
 for (K=1;K<=N;K++){
   TF0 = 0;
    TH0 = 0x0FE;               // Recarga Timers
    TL0 = 0xD4;                // para 1 mS  TH=FE TL=D4
    TR0 = 1;                   // Habilita TIMER 0
    while(!TF0);               // Espera sobrepasamiento
   }
  TR0 = 0;                     // Detiene Timer 0
 } // ------------------------------------------------------------


void LCD_INIT(){    // Inicializae el Modulo LCD Inteligente
  Dato = 0x3C;
  Send_Control();
} // ------------------------------------------------------------


void LCD_CLR(){      // Apaga Display
  Dato = 0x01;
  Send_Control();
} // ------------------------------------------------------------


void LCD_ON(){       // Limpia Display y Cursor Home
  Dato = 0x0C;
  Send_Control();
} // ------------------------------------------------------------


void MODO_LCD(){     // Modo Incremento sin Desplazamiento
  Dato = 0x06;
  Send_Control();
} // ------------------------------------------------------------
```

```
void DESP_CHAR(){  // Rutina que desplaza caracteres una posicion
  Dato = 0x18;
  Send_Control();
} // ------------------------------------------------------------

void LINEA_2(){   // Posicionar cursor en segunda Linea
  Dato = 0xC0;
  Send_Control();
} // ------------------------------------------------------------

// Rutina que genera las senales de CONTROL del LCD

void Send_Control(){
   RS = 0;                  //        ___
   E  = 1;                  //  RS      |____________________
   P1 = Dato;               //             ___
   Demora (1);              //  EN   _____|   |_____________
   E  = 0;                  //              ___
   Demora (1);              //  DATA _____////_______________
} // ------------------------------------------------------------

// Rutina que genera las senales de DATOS del LCD

void Send_Data(){
   RS = 1;                   //            ____________________
   E  = 1;                   //  RS    ___|
   P1 = Dato;                //               ___
   Demora (1);               //  EN    _____|   |_____________
   E  = 0;                   //               ___
   Demora (1);               //  DATA _____////_______________
} // ------------------------------------------------------------


// * * * *  Programa Principal * * * *

void main(void){
     Demora (50);            // Modulo LCD
     LCD_INIT();
     Demora (5);
     LCD_INIT();
     Demora (5);
     LCD_INIT();
     Demora (5);

     LCD_INIT();              // Inicializa LCD
     LCD_ON();                // Activa Display
     LCD_CLR();               // Limpia Display y Cursor a Home
     MODO_LCD();              // Programa Modo del LCD

     for (i=0;i<=31;i++)      // Envia Caracteres del Arreglo al LCD
     {
       if (i==15) LINEA_2();  // Control para pasar a Linea 2
       Dato = Mensaje[i];
       Send_Data();           // Envia Caracteres
     }
    for (;;);
}
```