# Capítulo 4

## Interfase Humana

Para poder interactuar con cualquier máquina con la cual se trabaje se necesita una interfase, a través de la cual el usuario puede saber cual es el status actual de la máquina recibiendo información del sistema, luego, en base a los deseos del usuario se interactúa por medio de la interfase para crear un cambio en el sistema.

El sistema de electroporación necesita leer señales, generar salidas e interactuar con el usuario haciendo posible la entrega de los pulsos a la célula. En este capítulo se verán algunos aspectos de la interfase con el sistema.

## 4.1 Teclado Matricial.

Los teclados matriciales son realmente una extensión del concepto de botón, son un simple arreglo de botones conectados en filas y columnas, figura "3.1", de modo que se pueden leer un gran número de botones de entrada con el número mínimo de terminales requeridos por el microcontrolador.

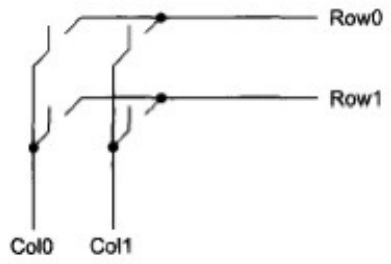


Figura 3.1 "Conexiones del Teclado"

Un teclado matricial 4×4 solamente ocupa 4 líneas de un puerto para las filas y otras 4 líneas para las columnas, de este modo se pueden leer 16 teclas utilizando solamente 8 líneas de un puerto del microcontrolador. Si se asume que todas las columnas y filas inicialmente están en alto (1 lógico), la activación de un botón se puede detectar al enviar por las líneas de salida (renglones) solo un (Ø lógico) por vez y sondear cada columna en busca de un cero, si ninguna columna está en bajo entonces se rota el Ø de las filas secuencialmente de tal manera que solamente un cero se encuentre entre los renglones del teclado cuando se realiza las lecturas de las líneas de entrada (columnas). Cuando el Ø llegue a la fila más significativa del teclado, debe reingresar en la próxima ocasión por la menos significativa, reiniciando la exploración del teclado.

El puerto B del microcontrolador viene preparado especialmente para el control de un teclado matricial 4×4. Para tener siempre un valor de voltaje alto (l lógico) en los renglones del teclado (parte alta del puerto B del µC) es necesario conectar los resistores de jalón figura "3.2", sin embargo el puerto B cuenta con resistores de jalón integradas, de ese modo es posible trabajar con un teclado matricial sin necesidad de ningún componente externo.

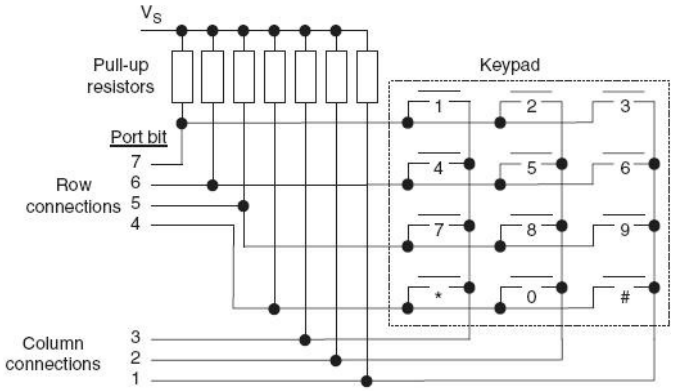


Figura 3.2 "Teclado, con resistores de Jalón"

Los resistores de jalón del puerto B se habilitan poniendo en o el bit RBPU del registro INTCON2. Al método expuesto para detectar la activación de una tecla en un teclado matricial se le conoce como muestreo secuencial. Existen otros, sin embargo este es tal vez el más sencillo.

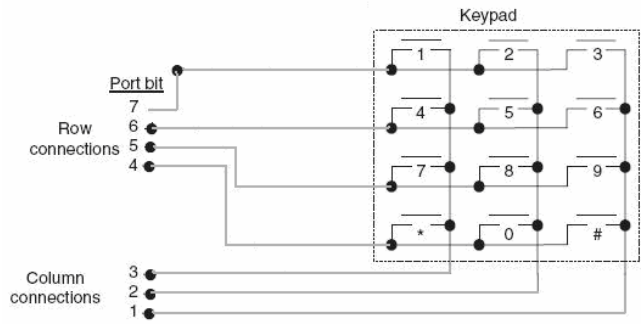


Figura 3.3 "Teclado, conectado al Puerto B resistores de jalón internas"

El programa principal es muy sencillo, simplemente configura la parte baja del puerto B como entradas y la parte alta del puerto B como salidas. Además de configurar el puerto también se habilitan los resistores de jalón internas con la línea bitclr (INTCON2,RBPU). Después de la configuración se llama a la rutina get_key () encargada de escanear el teclado y regresar el valor de la tecla pulsada a una variable para su posterior manejo. La rutina get_key () escanea el teclado matricial utilizando el método descrito.

```
void INI_keypad (void)
{
    bitclr (INTCON2,RBPU);          //Se activan las resistencias de Pull-Up del puerto B
    TRISB=0X0F;                  //(B0:B3) -->>Entradas  (B4:B7) -->> Salidas
    PORTB=0XF0;                  //Pone en alto las salidas.
}
```

Primero se carga un Ø a la primer fila, después se checa columna a columna si hay un Ø, en caso de que no se detecte ningún Ø, el Ø de las filas se recorre y se vuelven a revisar las columnas. Si no se detectó ningún Ø significa que no se presionó ninguna tecla y la rutina vuelve a comenzar. Si se detectó un Ø significa que se presionó una tecla y en ese momento se genera un retardo de tiempo en la lectura del teclado, de tal manera que se ignoren los contactos subsiguientes a los rebotes causados por la activación de la tecla. Después de eliminar los rebotes se decodifica la tecla pulsada utilizando una tabla de equivalencias y saber así, qué tecla fue pulsada.

```
char get_key (void)
{
    char renglon, columna, key;       //Variables para guardar la informacion de la Tecla
    char tecla;

    renglon=0xEF;
    PORTB=renglon;          //Inicio del testeo de las teclas por renglon
    while ((B0&&B1&&B2&&B3)!=0)
    {
        renglon<<=1;        //Pasa a revisar todas los renglones
        if (renglon == 0xf0)
        {
            key=0xFF;       //Devuelve que ninguna tecla se apreto
            return key;
        }
        else
            PORTB=renglon;
    }

    Delay10KTCYx (20);      //Debounce de 20 ms
    columna=PORTB;              //Guarda en columna el contenido de todo el puerto.
    renglon&=0xF0;          //Se guarda en renglon el nibble alto de tecla.
    columna&=0x0F;          //Se guarda en columna el nibble bajo de tecla.
    tecla=renglon|columna;
```

La función get_key sólo retornará cuando un tecla haya sido oprimida. Esta rutina no permite los rebotes dentro de las teclas. El código espera a que el botón pulsado sea liberado y hace un retardo de tiempo de 20 ms para eliminar los posibles rebotes que genera la tecla al ser liberada.

```
void debounce (void)                    //Pregunta si se sigue presionando la tecla.
{
    while ((B0&&B1&&B2&&B3)==0)
        {   }
    Delay10KTCYx (20);          //Debounce de 20 ms
}
```

Este método es ideal para aplicaciones que no tienen interrupciones y no se tiene necesidad de alguna aplicación mientras que la tecla es pulsada. Para resolver este problema se puede localizar la rutina get_key dentro una rutina de interrupción que se genera cada cierto tiempo.

## 4.2 Display de Cristal Líquido o LCD

Antes de mostrar la forma de conectar estos displays con el microcontrolador, se hará un pequeño recuento de las principales características que ellos tienen, las cuales servirán para entender mejor los programas y los diagramas que se muestran más adelante.


Figura 3.4 "Display LCD 2x16"

El controlador de LCD más común es el Hitachi 44780, el cual provee una conexión entre el µC y el LCD muy fácil de usar, poseen diferentes presentaciones, por ejemplo (2 líneas por 16 caracteres), 2x20, 4x16, 4x20, 4x40 etc, son de bajo costo y se consiguen fácilmente en el comercio. La forma de utilizarlos y sus interfaces son similares, por eso, los conceptos que se verán pueden ser empleados en cualquiera de ellos. Para el desarrollo del electroporador, se trabajó con un display de 4x16, ya que su tamaño es suficiente para la aplicación de despliegue de parámetros. La explicación del funcionamiento será basado en un LCD 2x16.

| Número de Terminal | Símbolo | Función |
|---|---|---|
| 1 | Vss | GND |
| 2 | Vdd | +3V ó +5V |
| 3 | Vo | Control de Ajuste |
| 4 | RS | H/L Registro de Selección |
| 5 | R/W | H/L Lectura/Escritura |
| 6 | E | Habilitación |
| 7 | DB0 | H/L Línea de Datos |
| 8 | DB1 | H/L Línea de Datos |
| 9 | DB2 | H/L Línea de Datos |
| 10 | DB3 | H/L Línea de Datos |
| 11 | DB4 | H/L Línea de Datos |
| 12 | DB5 | H/L Línea de Datos |
| 13 | DB6 | H/L Línea de Datos |
| 14 | DB7 | H/L Línea de Datos |
| 15 | A/Vee | Anodo |
| 16 | K | Catodo |

Tabla 3.1 "Configuración de terminales del módulo LCD HD44780"

La tabla 3.1 muestra la configuración de terminales que se encuentran comúnmente en un LCD, aunque su ubicación cambia, conforme al LCD que se este utilizando, por ese motivo, es recomendable ver las hojas de datos del fabricante, aunque la mayoría de los LCD conservan las mismas funciones. Algunos LCD tienen luz posterior o "backlight", para mejorar su visualización, ésta se maneja a través de dos terminales que normalmente se conectan a +5V y tierra por medio de una resistencia de 10 Ω para alimentar al positivo del backlight.

Las terminales de conexión de estos módulos incluyen un bus de datos de 8 bits, una terminal de habilitación (Enable), una Terminal de selección, que indica que el dato es una instrucción o un caracter del mensaje (Register Select) y una Terminal que indica si se va a escribir o leer en el módulo LCD (Read/Write).

Según la operación que se desee realizar sobre el LCD, las Terminales de control E, RS y R/W deben tener un estado determinado. Además, debe tener en el bus de datos un código que indique un caracter para mostrar en la pantalla o una instrucción de control. El módulo LCD responde a un conjunto especial de instrucciones, estas deben ser enviadas por el microcontrolador o sistema de control al display, según la operación que se requiera. En las tablas 3.2.a y 3.2.b se muestra el set de instrucciones y los bits de comandos respectivamente del LCD.

| Instruction | 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 display and returns cursor to the home position (address 0). | 1.64ms |
| Cursor home | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | * | Returns cursor to home position (address 0). Also returns display being shifted to the original position. DDRAM contents remains unchanged. | 1.64ms |
| Entry mode set | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | I/D | S | Sets cursor move direction (I/D), specifies to shift the display (S). These operations are performed during data read/write. | 40us |
| Display On/Off control | 0 | 0 | 0 | 0 | 0 | 0 | 1 | D | C | B | Sets On/Off of all display (D), cursor On/Off (C) and blink of cursor position character (B). | 40us |
| Cursor/display shift | 0 | 0 | 0 | 0 | 0 | 1 | S/C | R/L | * | * | Sets cursor-move or display-shift (S/C), shift direction (R/L). DDRAM contents remains unchanged. | 40us |

| Instruction | Code | | | | | | | | | | Description | Execution time |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RS | R/W | DB7 | DB6 | DB5 | DB4 | DB3 | DB2 | DB1 | DB0 | | |
| Function set | 0 | 0 | 0 | 0 | 1 | DL | N | F | * | * | Sets interface data length (DL), number of display line (N) and character font(F). | 40us |
| Set CGRAM address | 0 | 0 | 0 | 1 | CGRAM address | | | | | | Sets the CGRAM address. CGRAM data is sent and received after this setting. | 40us |
| Set DDRAM address | 0 | 0 | 1 | DDRAM address | | | | | | | Sets the DDRAM address. DDRAM data is sent and received after this setting. | 40us |
| Read busy-flag and address counter | 0 | 1 | BF | CGRAM / DDRAM address | | | | | | | Reads Busy-flag (BF) indicating internal operation is being performed and reads CGRAM or DDRAM address counter contents (depending on previous instruction). | 0us |
| Write to CGRAM or DDRAM | 1 | 0 | write data | | | | | | | | Writes data to CGRAM or DDRAM. | 40us |
| Read from CGRAM or DDRAM | 1 | 1 | read data | | | | | | | | Reads data from CGRAM or DDRAM. | 40us |

Tabla 3.2.a "Set de instrucciones del HD44789"

| Bit name | Setting / Status | |
|---|---|---|
| I/D | 0 = Decrement cursor position | 1 = Increment cursor position |
| S | 0 = No display shift | 1 = Display shift |
| D | 0 = Display off | 1 = Display on |
| C | 0 = Cursor off | 1 = Cursor on |
| B | 0 = Cursor blink off | 1 = Cursor blink on |
| S/C | 0 = Move cursor | 1 = Shift display |
| R/L | 0 = Shift left | 1 = Shift right |
| DL | 0 = 4-bit interface | 1 = 8-bit interface |
| N | 0 = 1/8 or 1/11 Duty (1 line) | 1 = 1/16 Duty (2 lines) |
| F | 0 = 5x7 dots | 1 = 5x10 dots |
| BF | 0 = Can accept instruction | 1 = Internal operation in progress |

Figura 3.2.b "Bit de comandos del HD44780"

La interfase entre el microcontrolador y el LCD se puede realizar con el bus de datos a 4 u 8 bits. Las señales de control trabajan de la misma forma en cualquiera de los dos casos, la diferencia se establece en el momento de iniciar el sistema, ya que existe una instrucción que permite seleccionar dicha configuración. Estas conexiones se explican más adelante de forma detallada.

Los caracteres que se envían al display se almacenan en la memoria RAM del módulo. Existen posiciones de memoria RAM, cuyos datos son visibles en la pantalla y otras que no lo son, estas últimas se pueden utilizar para guardar caracteres que luego se desplazan hacia la parte visible. En la figura "3.5" se muestran las direcciones de memoria visibles y no visibles, que conforman las dos líneas de caracteres del módulo.

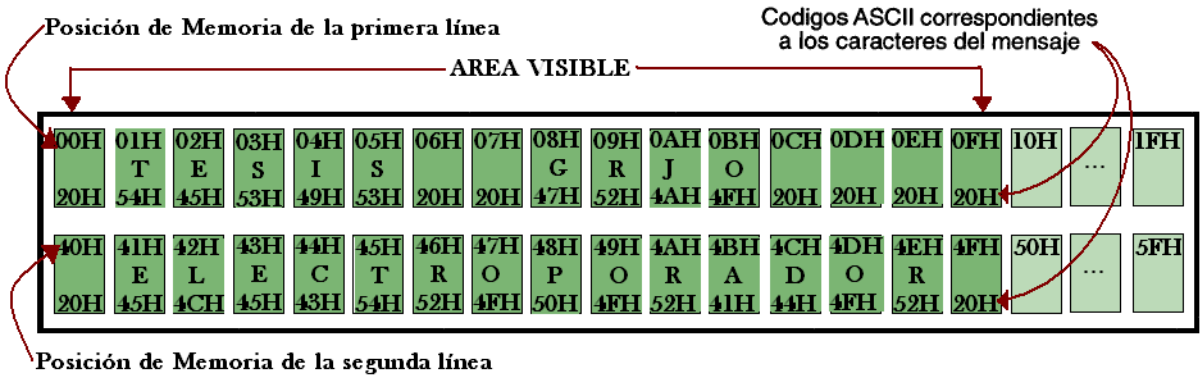


Figura 3.5 "Mapa de memoria del HD44780"

Es importante señalar que sólo se pueden mostrar caracteres ASCII de 7 bits, por lo tanto, algunos caracteres especiales no se pueden ver. Por otra parte, se tiene la opción de crear caracteres especiales (creados por el programador), y almacenarlos en la memoria RAM que posee el LCD como se muestra en la figura 3.6.

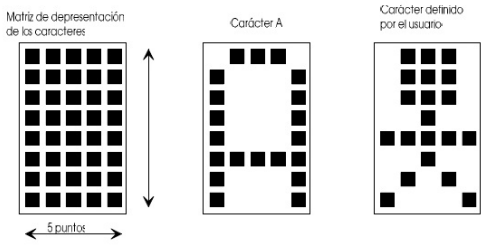


Figura 3.6 "Caracteres formados dentro del CGRAM"

### *4.2.1. Interfase con microcontrolador a LCD*

La conexión entre el PIC y el módulo LCD se realizará por medio de un bus de datos de 4 bits, donde se utilizarán las 4 Terminales de mayor peso del puerto D (RD4-RD7) del microcontrolador. Las señales de control (R/W, RS y E ), se generarán con las Terminales de menor peso del puerto D (RD1, RD2 y RD3). La figura 3.7 muestra esta configuración.

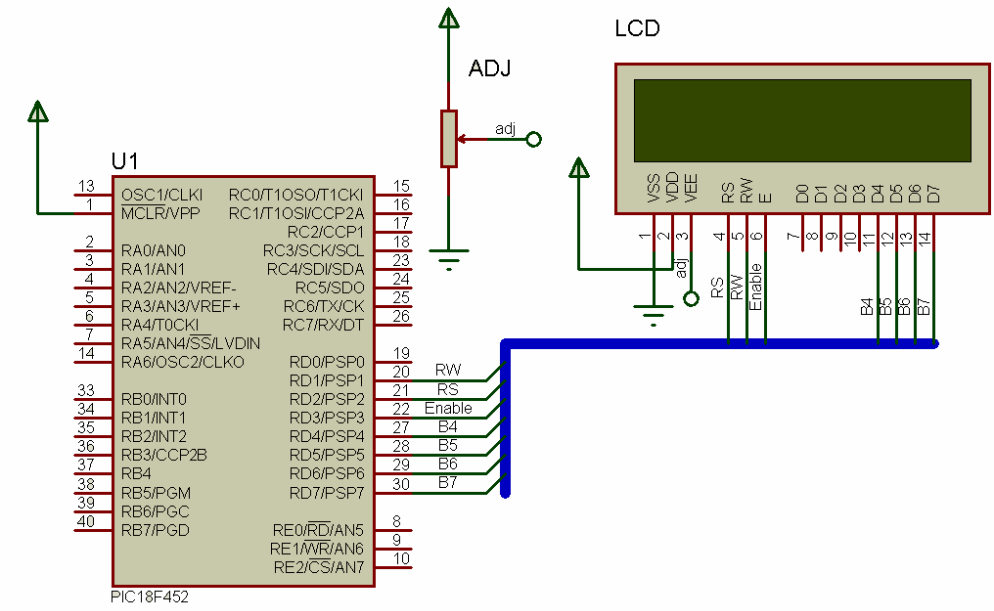


Figura 3.7 "Conexión a 4 bits del LCD con el microcontrolador "

El software que se implanto en el microcontrolador, se encarga de mostrar mensajes en un LCD de 4x16. Dichos mensajes ocuparan las 4 líneas de la pantalla y permanecen fijos, o cambiaran según sea necesario dentro del menú contextual en el cual se encuentre el usuario.

Los pasos más importantes para la inicialización del módulo LCD son:

- ✓ Se programan los puertos según el circuito.
- ✓ Se debe inicializar el módulo LCD. El primer dato que se envía (2C h) le indica al módulo que la comunicación se va a realizar a 4 bits, se emplearán todas las líneas de caracteres del LCD y el formato de la celda del LCD. El dato se puede deducir con la lista de las instrucciones que se muestra en la figura "3.5.a". La función de INI_LCD, se encarga de generar todas las señales de inicialización y los tiempos necesarios para que exista una correcta comunicación.

```
void INI_LCD(void)
{
   TRISD= 0;                  //Se elige como salida el puerto D
   PORTD=0;                       //Pone a cero el puerto D
   bitclr(PORTD,RS);          //Linea RS en bajo (PortD, bit2)
   Delay10KTCYx (100);            //Retardo de  100ms
   PUERTO_LCD |= BIT4 | BIT5;         //D7-D4 = 0011
   PUERTO_LCD &= ~BIT6 & ~BIT7;
   _E();                           //Activa Enable
   Delay10KTCYx (10);             //Retardo de 10ms
   _E();
   Delay10KTCYx (10);
   _E();
   Delay10KTCYx (10);
   PUERTO_LCD &= ~BIT4;
   _E();                           //toggle E
   LCD_COM(LINES_5X7);        //4 BITS , 2 LINEAS, 5X7
   LCD_COM(DISP_ON);
   LCD_COM(CLR_DISP);              //ESTE COMANDO TARDA MAS QUE LOS DEMAS
   temp=PUERTO_LCD;
   Delay10KTCYx (100);             //Delay 100ms
}
```

Las funciones LCD_COM y LCD_DAT que se muestran a continuación envían primero el nibble alto (4 bits de mayor peso) del dato y luego el nibble bajo (4 bits de menor peso). En el momento de enviar cada uno de los datos de 4 bits, las señales de control deben comportarse de la misma forma como si fuera un dato completo. Por eso en la rutina se ve que la señal RS conserva el nivel lógico adecuado y la señal E genera los dos pulsos que se requieren (el primero para el nibble alto y el segundo para el bajo).

```
void LCD_DAT ( char d)
{
    Delay1KTCYx (5);          //Retardo de 0.5ms  5000 ciclos@40 MHZ
    temp = d & 0xf0;                  //ENVIA NIBBLE SUPERIOR
    PUERTO_LCD &= 0x0f;
    PUERTO_LCD |= temp;
    bitset(PORTD,RS);            //ENVIA A REGISTRO DE DATOS
    _E();                        //Activa Enable
    temp = d & 0x0f;
    temp = temp << 4;            //ENVIA NIBBLE INFERIOR
    PUERTO_LCD &= 0x0f;
    PUERTO_LCD |= temp;
    bitset(PORTD,RS);            //ENVIA A REGISTRO DE DATOS
    _E();                        //TOGGLE E
}


 void LCD_COM ( char c)
 {
     Delay10KTCYx (10);             //10ms , 100000CICLOS A 40 Mhz.
     temp = c & 0xf0;      //get upper nibble
     PUERTO_LCD &= 0x0f;
     PUERTO_LCD |= temp;            //send CMD to LCD
     bitclr(PORTD,RS);              //set LCD to CMD mode
     _E();                  //toggle E for LCD
     temp = c & 0x0f;
     temp <<= 4;            //get down nibble
     PUERTO_LCD &= 0x0f;
     PUERTO_LCD |= temp;
     bitclr(PORTD,RS);          //set LCD to CMD mode
     _E();                  //toggle E for LCD
 }
```