Práctica 5. Diseño de una Cerradura Electrónica.

Guillermo Carpintero del Barrio
Susana Patón Álvarez

Esta práctica plantea el desarrollo de una aplicación, de forma que puedas orientar tu trabajo hacia la resolución de un problema concreto: desarrollar una cerradura electrónica. Esta cerradura consta de un teclado, mediante el cual se introduce un código. En función del código introducido, se genera una señal de acceso.

Para solventarlo, vas a necesitar aplicar técnicas que implican manejar los siguientes conceptos:

1) Decodificación del código de tecla mediante tablas almacenadas en ROM,

2) Interfase entre el microcontrolador y un teclado pasivo

3) Manejo de una pantalla LCD

4) Temporización

5) Interrupciones

Dos de los componentes básicos para esta práctica se suministrarán en el laboratorio, el teclado y la pantalla LCD.

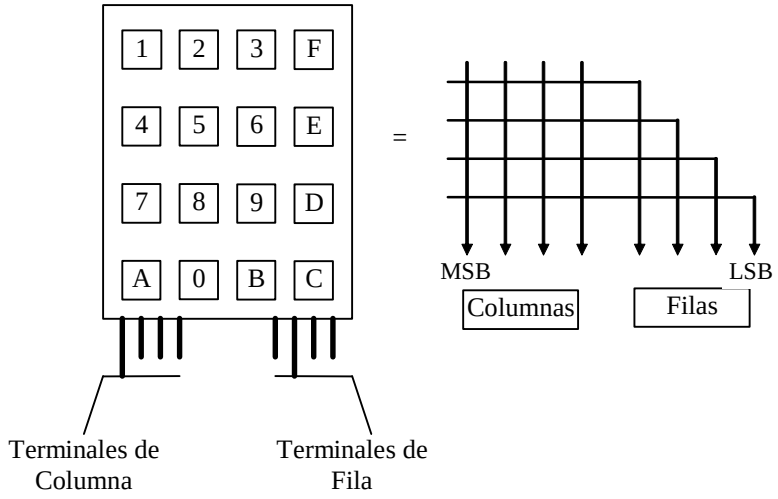
Hemos asignado dos sesiones a esta práctica de forma que durante la primera sesión trabajes con el teclado y aprendas diferentes formas de temporizar un evento. La segunda esta prevista para unirle el manejo de la pantalla LCD, visualizando las teclas pulsadas.

Cada sesión se ha dividido en tres pasos que deberá mostrar a su profesor de prácticas.

2.1 Sesión 1. Lectura de un teclado

Como te hemos indicado, esta sesión de dedica al interfase con un **teclado pasivo**. Este tipo de teclados que te proponemos descodificar se encuentra habitualmente en equipos electrónicos como interfase de entrada.

El teclado concreto del que dispones en tu puesto es un teclado pasivo **matricial 4x4,** el cual dispone de cuatro pines de interfase, a través de los cuales debemos identificar cuál de las dieciséis teclas de que dispone ha sido pulsada. El esquema de conexionado de este tipo de teclados se presenta en la siguiente figura:



Como puedes observar, de los ocho terminales que dispone el teclado, cuatro dan acceso a las columnas (pines 7 a 4), y cuatro dan acceso a las filas (pines 3 a 0).

Es importante destacar que el hecho de pulsar una tecla sólo implica que se cortocircuita la línea de la fila con la línea de la columna correspondiente. Es por ello que el microcontrolador debe tomar un papel activo en la lectura del teclado, realizando las siguientes tareas:

- Identificar la tecla pulsada
- Identificar pulsaciones individuales
- Evitar los 'rebotes'

Existen varias formas de conectarlo a un microcontrolador para su descodificación, sin embargo, para esta práctica te recomendamos que sacrifiques uno de los puertos de tu microcontrolador para la gestión del teclado. Elige un puerto, y conecta las 8 líneas del teclado a este.

Si ya has elegido el puerto, sigue leyendo. Si no, acude al Data Sheet de tu micro.

¡Bien! Pues ahora empieza la diversión. Debes pensar un algoritmo para identificar la tecla pulsada (te recordamos que eso implica determinar la fila y columna que han sido cortocircuitadas mediante la pulsación). **NOTA**: *Hay varias técnicas para ello, y te recomendamos que consultes bibliografía antes de ponerte a escribir código. Si no entiendes alguna de las opciones que tienes a tu disposición para hacerlo puedes preguntar a los profesores, pero ¡busca tu información para comenzar a trabajar!*

La gestión del teclado como ves implica realizar una primera función que es determinar si hay una tecla pulsada. En caso afirmativo, identificar la pulsación. Te recomendamos que estructures el código en base a funciones de C. De esta forma, cada puesto de prácticas debe crear una biblioteca de funciones. La biblioteca estará compuesta por un fichero de cabecera y varios ficheros fuente. Cada fichero fuente contendrá una sola función y se nombrará como dicha función. Estas son las funciones mínimas que debe tener su biblioteca:

- *LeeTecla*

  Prototipo: char LeeTecla(void);

  Es una función que deberá proporcionar el código de la tecla que se ha pulsado. En caso que de no se haya pulsado ninguna tecla deberá proporcionar el valor 0xF0.

  Los códigos de cada tecla deben estar almacenados en la memoria de programas, y son declarados e inicializados de forma global, es decir, de forma que su contenido es accesible a todas las funciones.

- *EsperaTecla*

  Prototipo: char EsperaTecla(void);

  La tarea de esta función es que no devuelva el control hasta que no se haya pulsado una tecla.

- *Tecla2Char*

  Prototipo: char Tecla2Char(char tecla);

  Esta función mapea el teclado, es decir, realiza la función de interfase entre los códigos del teclado y la tecla asociada que se le quiera colocar.

El mapeo se realizará en base a una tabla almacenada en la memoria de programas, cuyos elementos estén ordenados de la misma forma que la tabla de códigos.

El fichero de cabecera deberá contener los prototipos de estas funciones y las dos tablas que se necesitan, la de códigos y la de mapeo de caracteres.

***RECOMENDACIÓN**: Estructura la función LeeTecla de forma que la lectura del puerto en el que se conecta el teclado devuelva una palabra de 8 bits, formado por dos palabras de 4 bits, que codifican las coordenadas de fila y columna de la tecla que se está pulsando.*

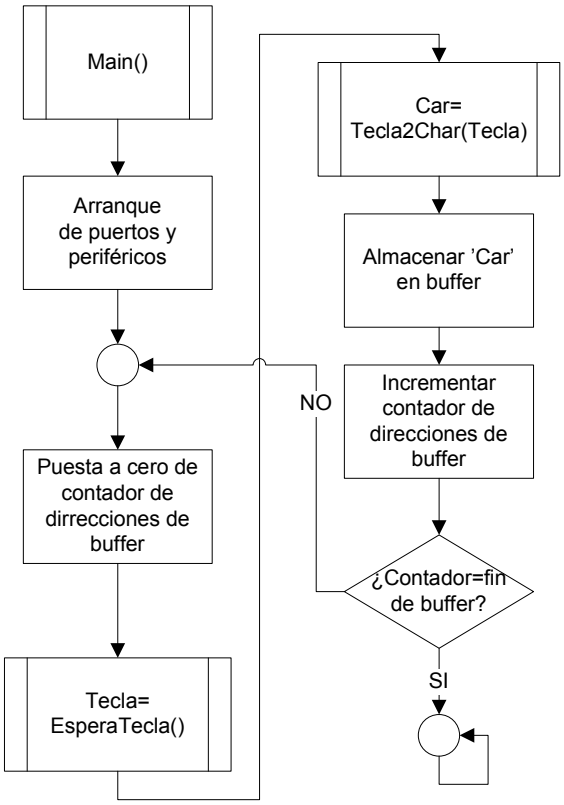
Normalmente la primera de estas palabras (4 bits MSB) nos proporciona la coordenada de la fila y la segunda (4 bits LSB) la columna, sin más que imponer un cierto valor lógico en uno de los extremos de los contactos, esto es, en una fila o en una columna. Para evitar ambigüedades, lo habitual es escoger lógica activa a nivel bajo, de forma que cuando no hay ninguna tecla pulsada, los 8 bits están a nivel alto.

Una tecla pulsada provoca la aparición de un cero en los bits correspondientes a su coordenada en el teclado. Esto es, aparece un cero en una posición de cada una de las palabras de fila y columna según se presenta en la siguiente tabla.

0	1011 1110	8	1011 1101
1	0111 0111	9	1101 1101
2	1011 0111	A	0111 1110
3	1101 0111	B	1101 1110
4	0111 1011	C	1110 1110
5	1011 1011	D	1110 1101
6	1101 1011	E	1110 1011
7	0111 1101	F	1110 0111

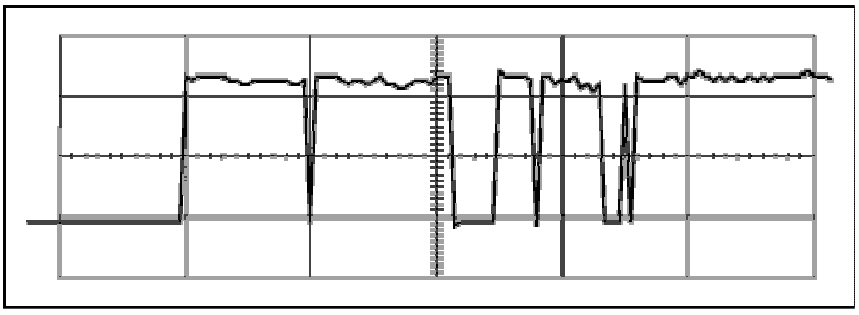
Paso 1.

Diseñar, y depurar en el circuito, un programa que espere 4 teclas y las guarde en una zona de memoria reservada para este propósito. Con ayuda del 'debugger' visualice las 4 posiciones de memoria. El programa deberá tener el siguiente diagrama de flujo:



Paso 2. Incorporando una Función Antirrebotes

Los interruptores, al pulsarse, no generan señales cuadradas perfectas de cambio de estado en la linea al pulsarlos. Realmente, la señal en la línea aparece como se presenta en la figura siguiente.



Aparecen unos transitorios a partir del instante de pulsación, observandose múltiples transiciones (denominados rebotes) durante un intervalo de unos 5 ms. Tras este intervalo inicial, la línea permanece en un estado estable. Para evitar cualquier influencia de estos transitorios en la lectura e identificación de la tecla, nuestro programa debe incorporar una función anti-rebotes.

Modifique la función EsperaTecla() de forma que cuando se detecte una tecla pulsada, ocurra una espera de 20ms y se lea de nuevo el teclado. Si las dos lecturas consecutivas coinciden se dará la tecla por válida, devolviendo el

valor de la tecla pulsada. Si las lecturas no coinciden la función continuará leyendo el teclado cada 20ms hasta que dos lecturas consecutivas coincidan. La temporización de 20ms se realizará con la ayuda de una de las interrupciones de los timers. Diseñar y depurar en el circuito el programa del paso anterior con la modificación realizada.

Paso 3. Identificación de un Código PIN.

Una vez que estamos seguros de que nuestro programa de lectura del teclado identifica las teclas sin problemas, pasamos ahora a considerar el hecho de que en la aplicación, se deben leer cuatro teclas consecutivas.

Modifique el programa principal de forma que se comparen las cuatro cifras introducidas con el código prefijado 84A1, almacenado en ROM. Si coinciden ambos códigos, el introducido por el usuario y el almacenado en memoria, deberá encenderse el LED de la placa. En caso contrario, se esperará un nuevo código de entrada. El LED de la placa deberá estar siempre apagado al comienzo e la secuencia.

Cuestiones teóricas de la sesión 2.1

1. ¿Qué puerto utilizará para conectar el teclado y por qué? ¿Que registros necesita configurar para usar dicho puerto?

2. Describa el método que va a usar para leer el teclado y dibuje el diagrama de flujo correspondiente a la rutina LeeTecla()

3. Escriba el fichero de cabecera de la nueva biblioteca

4. ¿Cuál es el timer que ha elegido para el paso 2? ¿Qué registros tiene que configurar y con qué valores? Justifique los cálculos

5. Dibuje y comente el diagrama de flujo de la rutina EsperaTeccla() del paso 2, junto con el diagrama de flujo de la rutina de atención a la interrupción.

6. Dibuje y comente el diagrama de flujo de la función main() del paso 3.

7. Imprima un listado del código fuente del paso 3 comentado.

8. Después de compilar y lincar el programa del paso 3, ¿Qué contenido hay en la dirección 8 de la memoria de programa? Coméntelo

2.2 Sesión 2. Uso de una Pantalla LCD

Esta sesión se dedica a la interfase con la pantalla LCD.

Las siglas LCD se refieren a Liquid Crystal Display o Pantalla de Cristal Líquido. Básicamente es una matriz de puntos que, según se indique de forma electrónica, toma una tonalidad más oscura que el fondo, o de la misma que él. El manejo de la matriz de puntos del LCD es extremamente complejo, por lo que es necesario un controlador específico que sirva de interfaz entre la información que el Sistema Digital quiere mostrar, y los píxeles que deben ser activados. El controlador más extendido del mercado es el HITACHI 44780. La representación en pantalla se realiza mediante el mapeado de un juego de caracteres que se organizan en filas y columnas.

El LCD que usaremos en el laboratorio es un módulo de 2x16 caracteres, lo que significa que el LCD tiene 2 líneas de 16 caracteres cada una. Los caracteres que están representados en nuestro display de 2x16 se encuentran mapeados en una zona de memoria interna, de forma que lo que se ve en pantalla corresponde a una zona de memoria, según la siguiente tabla:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	Posición LCD
Línea 1	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Dirección memoria (hex)
Línea 2	40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F	Dirección memoria (hex)

Uso de la biblioteca *xlcd* de MPLAB C18

Para facilitarnos el desarrollo de aplicaciones que hagan uso de periféricos como un LCD, el compilador dispone de bibliotecas de funciones.

La biblioteca de funciones que nos ofrece el compilador C18 de Microchip aparece en el documento "MPLAB C18 Compiler Libraries". En este documento se nos indica (ver el índice) que disponemos de una librería de periféricos denominada **External LCD Functions**.

Las funciones de que disponemos son las siguientes:

TABLE 3-1: EXTERNAL LCD FUNCTIONS

Function	Description
BusyXLCD	Is the LCD controller busy?
OpenXLCD	Configure the I/O lines used for controlling the LCD and initialize the LCD.
putcXLCD	Write a byte to the LCD controller.
putsXLCD	Write a string from data memory to the LCD.
putrsXLCD	Write a string from program memory to the LCD.
ReadAddrXLCD	Read the address byte from the LCD controller.
ReadDataXLCD	Read a byte from the LCD controller.
SetCGRamAddr	Set the character generator address.
SetDDRamAddr	Set the display data address.
WriteCmdXLCD	Write a command to the LCD controller.
WriteDataXLCD	Write a byte to the LCD controller.

Estas funciones nos permiten realizar operaciones tan diversas como inicializar el LCD, escribir caracteres o configurar el número de líneas del display. La descripción de cada una de estas fuciones se puede encontrar tanto en el documento mencionado, como en el código fuente de la librería, **xlcd.h**.

Se recomienda encarecidamente que consultes la documentación disponible de la librería. Fundamentalmente, porque hay que tener en cuenta dos puntos importantes al hacer uso de esta:

1.- Funciones necesarias no incluidas

La función de inicialización del LCD necesita tres funciones que no están incluidas en la biblioteca, y que son necesarias para realizar las diferentes esperas. Estas funciones son:

```
void DelayFor18TCY( void );  //Espera de 18 ciclos máquina

void DelayPORXLCD (void); // Espera de 15ms

void DelayXLCD (void) // Espera de 5ms
```

Si miras el fichero xlcd.h, verás que esto se indica en la cabecera:

```
*       - The user must provide three delay routines:
*           - DelayFor18TCY() provides a 18 Tcy delay
*           - DelayPORXLCD() provides at least 15ms delay
*           - DelayXLCD() provides at least 5ms delay
```

Así que en el código fuente del programa debes escribir tres rutinas, con los nombres que se te ha indicado, y que generen las temporizaciones pedidas. Para la implementación de estas funciones existen varias opciones:

- usar bucles for, como los vistos en la práctica 1
- usar un timer con una rutina de espera activa, sin usar interrupciones
- usar las funciones de la biblioteca *delay*
- usar instrucciones Nop() (que consumen 1 ciclo máquina)

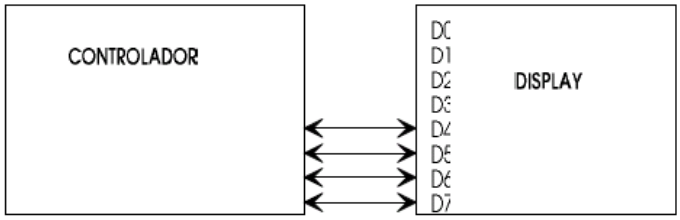
2.- Puerto de salida de la libreria

La librería asume que el puerto al que está conectado el LCD es el puerto B. De hecho, si consultas el fichero xlcd.h, encontrarás la siguiente información al respecto:

```
/* DATA_PORT defines the port to which the LCD data lines are connected
*/
#define DATA_PORT      PORTB
#define TRIS_DATA_PORT TRISB


/* CTRL_PORT defines the port where the control lines are connected.
 * These are just samples, change to match your application.
 */
#define RW_PIN   PORTBbits.RB6   /* PORT for RW */
#define TRIS_RW  DDRBbits.RB6    /* TRIS for RW */
#define RS_PIN   PORTBbits.RB5   /* PORT for RS */
#define TRIS_RS  DDRBbits.RB5    /* TRIS for RS */
#define E_PIN    PORTBbits.RB4   /* PORT for E  */
#define TRIS_E   DDRBbits.RB4    /* TRIS for E  */
```

Como en la placa de trabajo, el puerto B está ya dedicado a la función de Debugger, debemos cambiar la librería, para cambiar el puerto de interfase con el LCD a otro en el cual dispongamos de ocho bits (cuatro bits para las señales de control, y cuatro para las señales de datos). Atención a la conexión con cuatro bits de datos entre el puerto del microcontrolador y el display LCD, que debe atender al esquema que muestra la figura siguiente.



Bus de 4 bits

Para modificar la librería deben seguirse una serie de pasos:

Rebuild library file

1. Create project folder to rebuild the C18 library
2. Copy the header file (from \mcc18\h\xlcd.h) and the source files (from \mcc18\src\traditional\pmc\XLCD*.*) for XLCD to the project folder
3. Create a project in MPLAB to rebuild the library
4. Select the device type for the project
5. Select the project language toolsuite as Microchip C18
6. Select from menu "Project->Build Options...->Project" and select Build library target radio button in MPASM/C17/C18/Suite tab
7. Add all the source code files in the Source Files of the project window
8. Add the header file xlcd.h in the Header Files of the project window
9. Build the project
10. Copy the p18f*xxx*.lib file from \mcc18\lib folder to the project folder (where *xxx* is the device type)
11. Create a file called mknewlib.bat in the project folder and put the following line in the file
    for %%i in (*.o) do mplib /r .\p18f*xxx*.lib %%i
12. Double click on the mknewlib.bat will replace the xlcd library functions in p18f*xxx*.lib with the new build

Paso 1.

Realice un programa que inicialice la pantalla LCD y escriba el mensaje 'Hola mundo' en la primera línea. La inicialización se hará con una línea de escritura y usando 4 bits para el bus de datos. Al final de la inicialización debe observar el cursor parpadeando en la primera posición de la pantalla.

Paso 2.

Modifique el programa del paso 3 de la sesión 2.1 de forma que cada tecla pulsada y considerada como válida por la función EsperaTecla() se represente en la primera línea del LCD.

Paso 3.

Modifique el programa anterior para completar la cerradura electrónica. El sistema mostrará un mensaje de bienvenida, y esperará a que se introduzcan las 4 cifras del código. Si el código es correcto se mostrará un mensaje en la pantalla que lo indique, al tiempo que se enciende el LED de la placa. Si el código no es correcto se mostrará un mensaje en el LCD indicándolo y el programa quedará a la espera de un nuevo código.

Cuestiones teóricas de la sesión 2.2

1. Indique mediante un esquema la conexión entre el LCD y el microcontrolador. ¿Qué pines va a destinar a esta conexión?

2. ¿Qué registros necesita configurar para usar esos pines?

3. Indique las modificaciones que debe hacer en el fichero xlcd.h para poder usar las funciones de la biblioteca

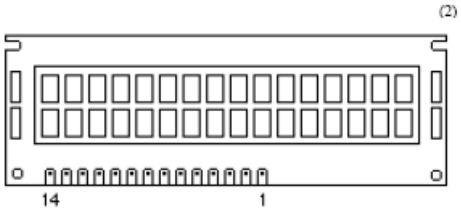
4. Escriba la implementación de las 3 funciones DelayFor18TCY(), DelayPORXLCD(), y DelayXLCD()

5. Dibuje el diagrama de flujo del paso 3.

6. Imprima el código fuente del paso 3 comentado.

ANEXO 1. Introducción a las pantallas LCD

La mayor parte de los displays LCD incorporan el controlador 44780 de Hitachi, y presenta el siguiente interfase con 14 pines de conexión al exterior:



Pin No	Name	I/O	Description
1	Vss	Power	GND
2	Vdd	Power	+5v
3	Vo	Analog	Contrast Control
4	RS	Input	Register Select
5	R/W	Input	Read/Write
6	E	Input	Enable *(Strobe)*
7	D0	I/O	Data *LSB*
8	D1	I/O	Data
9	D2	I/O	Data
10	D3	I/O	Data
11	D4	I/O	Data
12	D5	I/O	Data
13	D6	I/O	Data
14	D7	I/O	Data *MSB*

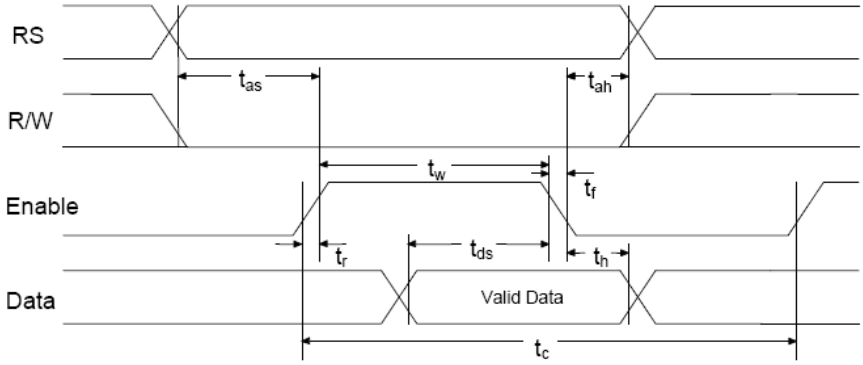
La comunicación de datos con el LCD se hace mediante un protocolo paralelo síncrono de 8 bits de datos y 3 líneas de control (E, R/W y RS), mientras que la alimentación y el control del contraste son las restantes.

A traves de las líneas de comunicación podemos transferir al controlador 44780 bien comandos (para seleccionar el formato de representación o realizar acciones especiales, como el borrado), bien caracteres (datos que deben presentarse en pantalla). Las líneas tienen la función siguiente:

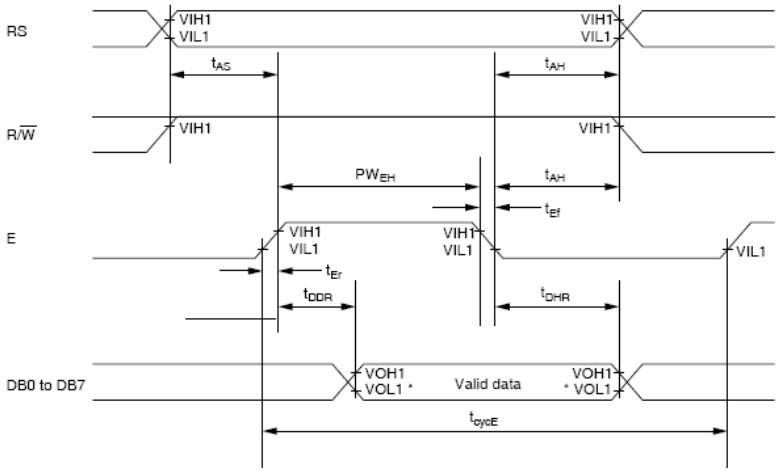
> La **señal RS** es la que se emplea para distinguir si en el bus de datos se encuentra un comando (mediante un '0') o un carácter (mediante un '1').
>
> La **señal R/W** indica si se desea hacer una operación de escritura o de lectura sobre el LCD, un '0' significa que el dato fluye hacia el LCD, mientras que un '1' significa que el dato sale del LCD.
>
> La **señal E** corresponde a la habilitación de los datos. El dato que se encuentra en el bus es validado con el flanco de bajada de esta señal.

Las tres señales de control deben ser impuestas por el microcontrolador siguiendo la temporizacion que especifican el fabricante del dispositivo. Para ambos tipos de transferencias (mandar un comando o un carácter al LCD), esta

temporización se especifica mediante el cronograma que adjuntamos. Os recomendamos acudir al catálogo del controlador para conocer los tiempos que aparecen en la figura.



Cada vez que el LCD recibe un comando o un carácter para representar, permanece ocupado durante un tiempo, durante el cual no es capaz de responder a ningún otro comando. El controlador dispone de un comando que permite saber si el LCD está ocupado o no. En este caso los datos fluyen del LCD al microcontrolador según el siguiente cronograma (consultar el catálogo del controlador para conocer los tiempos que aparecen en la figura):



En la siguiente tabla aparece un resumen de los diferentes comandos que se pueden enviar al LCD. Para más información debe consultar el catálogo del controlador.

Instruction	RS	RW	D7	D6	D5	D4	D3	D2	D1	D0	Description	Clocks
NOP	0	0	0	0	0	0	0	0	0	0	No Operation	0
Clear Display	0	0	0	0	0	0	0	0	0	1	Clears display & sets address counter to zero.	165
Cursor Home	0	0	0	0	0	0	0	0	1	0	Sets address counter to zero, returns shifted display to original position. DDRAM contents remains unchanged.	3
Entry Mode Set	0	0	0	0	0	0	0	1	I/D	S	Sets cursor move direction, and specifies automatic shift.	3
Display Control	0	0	0	0	0	0	1	D	C	B	Turns display (D), cursor on/off (C) or cursor blinking(B).	3
Cursor/display shift	0	0	0	0	0	1	S/C	R/L	0	0	Moves cursor and shift display. DDRAM contents remains unchanged.	3
Function Set	0	0	0	0	1	DL	N	M	G	0	Sets interface data width(DL), number of display lines (N,M) and voltage generator control (G).	3
Set CGRAM Addr	0	0	0	1	Character Generator RAM						Sets CGRAM Address	3
Set DDRAM Addr	0	0	1	Display Data RAM Address							Sets DDRAM Address	3
Busy Flag & Addr	0	1	BF	Address Counter							Reads Busy Flag & Address Counter	0
Read Data	1	0	Read Data								Reads data from CGRAM or DDRAM	3
Write Data	1	1	Write Data								Writes data from CGRAM or DDRAM	3

Hay que destacar que existen dos modos de comunicación con el LCD: **con 4 bits o con 8 bits**. En el modo de comunicación a 4 bits, sólo se usan los 4 MSB del bus de datos, del D7 al D4, y todo comando y carácter se envía usando dos veces los cronogramas mostrados, una primera vez para enviar la parte alta del byte, y una segunda vez para enviar la parte baja del byte.

Antes de enviar un comando al LCD, es necesario inicializarlo adecuadamente. Para ello es necesario seguir los diagramas de flujo que aparecen en el catálogo del controlador. En esta sesión práctica se usará una interfase de 4 bits, por lo que la inicialización debe ser: