PERIFÉRICOS DE E/S

Teclados

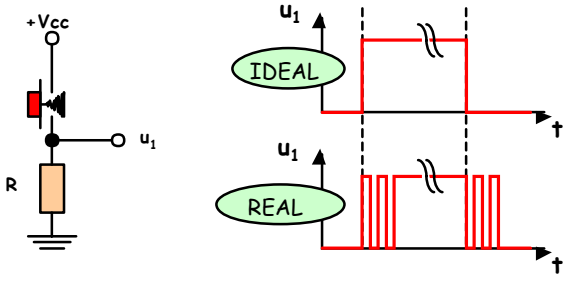
Muy utilizado para introducir información al microcontrolador.

Los hay de varios tipos: de lámina flexible, de efecto Hall, de efecto inductivo, de efecto capacitivo.

Los más comunes son los de lámina flexible.

El problema de los rebotes.

Debido al efecto muelle del pulsador, se producen oscilaciones en la señal tanto al pulsar como al soltar la tecla.



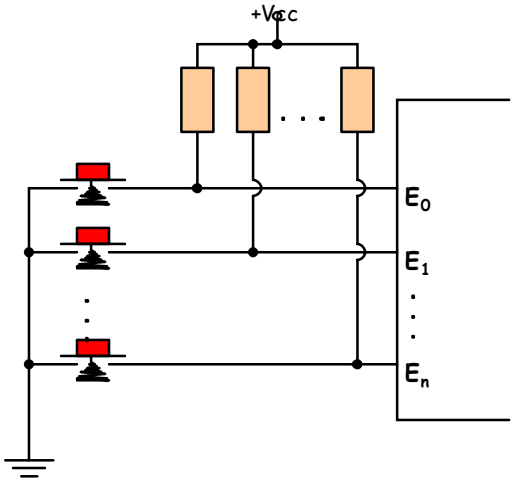
SOLUCIONES

Hw: Red R-C Biestables

Sw: Espera de un tiempo suficiente

Teclados lineales

Muy sencillos, pero no permiten disponer de muchas teclas.

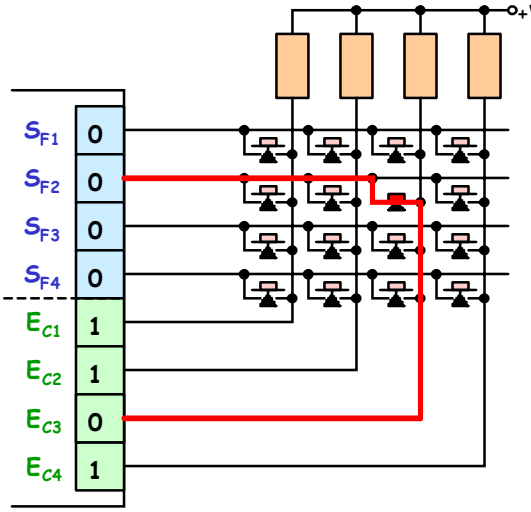


De este modo, cuando el microcontrolador detecte un "0" al final de la línea, se sabrá que se ha pulsado una tecla y, además, se sabrá cuál ha sido.

Basta con que el programa compruebe periódicamente el estado de las entradas a las que se ha conectado el teclado.

Teclados matriciales

Varias teclas controladas con un número reducido de puertos E/S.



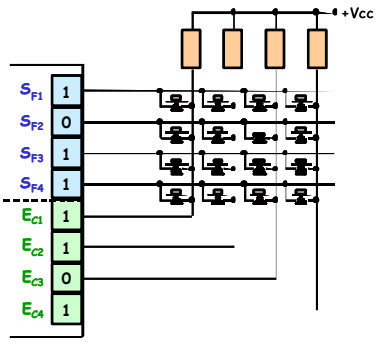
La pulsación de una tecla se pone de manifiesto en las entradas del microcontrolador conectadas al teclado.

En este ejemplo, se sabe que se ha pulsado una tecla de la tercera columna, pero no se sabe cuál.
Se necesita desarrollar algoritmos que permitan determinar cuál es la tecla que se ha pulsada.

Muestreo secuencial.

Una vez que se ha detectado que se ha pulsado una tecla, se cambia el valor de las salidas en el microcontrolador de modo que sólo una de ellas valga '0' en cada instante.

La combinación que dé lugar a un '0' en alguna de las entradas identificará la tecla que se ha pulsado.

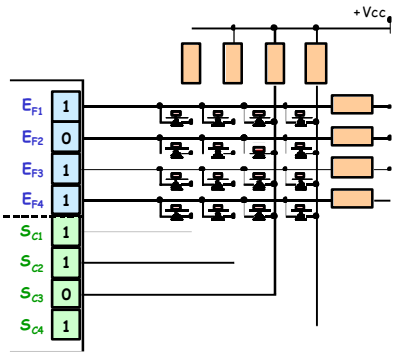


S_{F1}	S_{F2}	S_{F3}	S_{F4}	E_{C1}	E_{C2}	E_{C3}	E_{C4}
0	1	1	1	1	1	1	1
1	0	1	1	1	1	0	1
1	1	0	1	1	1	1	1
1	1	1	0	1	1	1	1

Es un método sencillo de implementar, pero tardará más o menos en encontrar la tecla pulsada en función de la posición que ocupe ésta.

Inversión de línea.

Tras detectar que hay una tecla pulsada, se almacena el valor que hay en las entradas, se invierten las líneas (las que eran entradas pasan a ser salidas y viceversa) y se saca por las nuevas salidas la combinación almacenada.



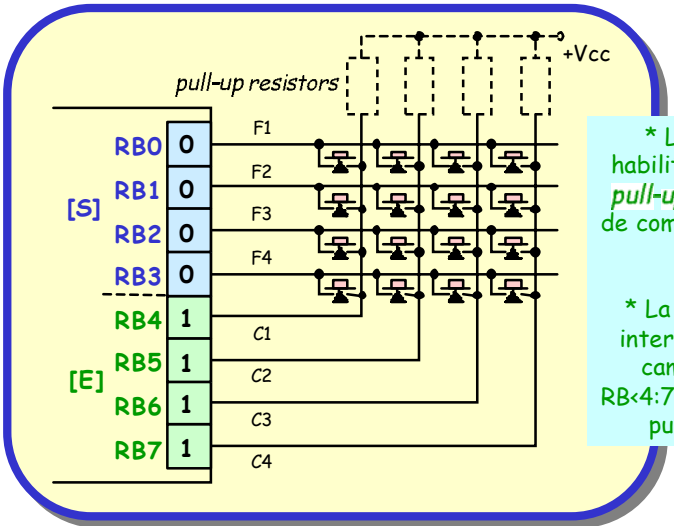
Esto dará lugar a que en las nuevas entradas sólo aparezca un cero en la fila a la que pertenece la tecla pulsada.

Como todos los bits son entrada en algún momento, se necesita colocar resistencias en los 8 bits.

Este método es más rápido que el anterior y tarda lo mismo en identificar cualquier tecla.

Conexión de teclados matriciales en los PIC.

El puerto B de los microcontroladores PIC está especialmente pensado para conectar un teclado matricial de 4×4.

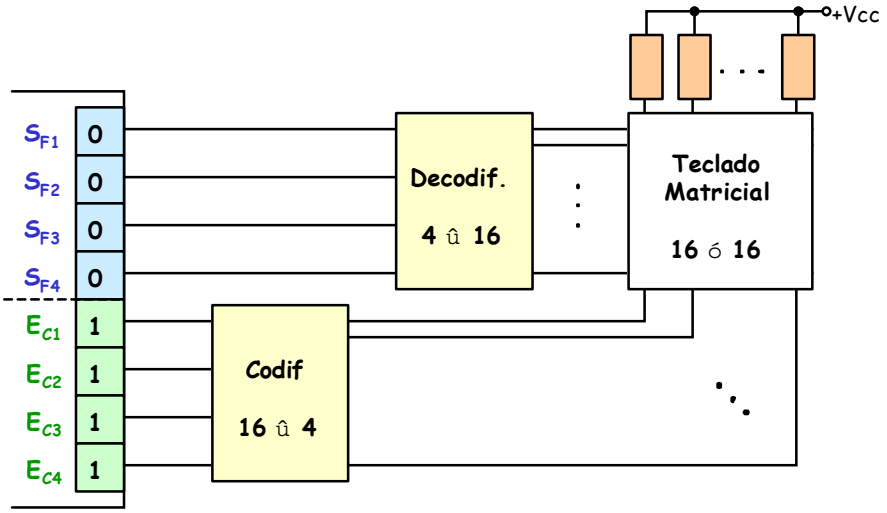


* La posibilidad de habilitar resistencias de *pull-up* reduce el número de componentes externos.

* La existencia de una interrupción asociada a cambios en los bits RB<4:7> avisa de que se ha pulsado una tecla.

Extensión a teclados de más de 16 teclas.

Se puede ampliar el tamaño del teclado sin más que utilizar codificadores y decodificadores.



Pantallas de cristal líquido (LCD)

Usado para representar caracteres alfanuméricos.

Control bastante complejo.

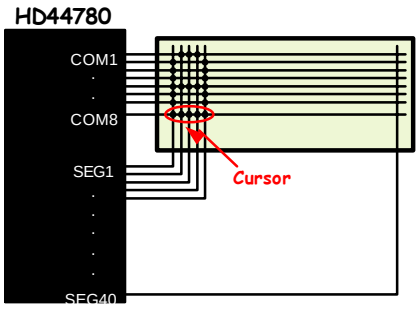
El control directo de los electrodos del LCD casi necesitaría un microcontrolador dedicado exclusivamente a esta tarea.

Lo más habitual es utilizar un LCD con un driver específico: el HD44780 de Hitachi o compatible.

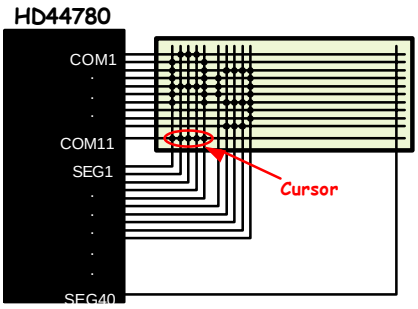
Características del HD44780

Driver para LCD de matriz de puntos para representación de caracteres y símbolos en formato 5ó8 ó 5ó10.

Dispone de 240 patrones de caracteres almacenados en ROM, de los cuales 208 de tamaño 5×8 y 32 de tamaño 5×10.



Ejemplo en 5×8 y 8 caracteres/línea



Ejemplo en 5×10 y 8 caracteres/línea

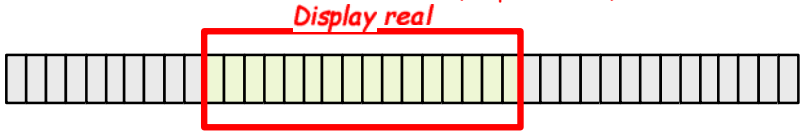
Memoria RAM de pantalla (Display Data RAM: DDRAM) con un total de 80 posiciones ó 8 bits/posición.

En los 8 bits se almacena el código del carácter para un generador de caracteres ROM que dispone de 240 caracteres posibles y 8 posiciones (dobles) para caracteres definibles por el usuario en una memoria CGRAM (caracteres gráficos).

Visibles 1 ó 2 líneas con 16 caracteres/línea.

Ventana de caracteres visibles (16 posiciones)

Display real



Registro: 1 2 ..... 11 ..... 26 ..... 39 40

Memoria de pantalla para una línea (40 posiciones)

Display virtual

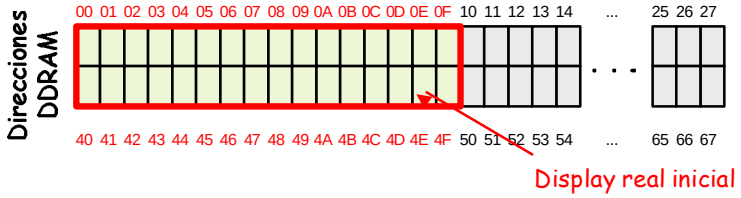
La DDRAM almacena el código de los caracteres que están siendo visualizados o que se encuentran en posiciones no visibles debido a la posición de la ventana de visualización.

Tiene un tamaño de 2 líneas ó 40 bytes/línea = 80 bytes.

Direcciones no contiguas entre las líneas 1 y 2.

0x00 a 0x27: 40 caracteres de la línea 1.

0x40 a 0x67: 40 caracteres de la línea 2.



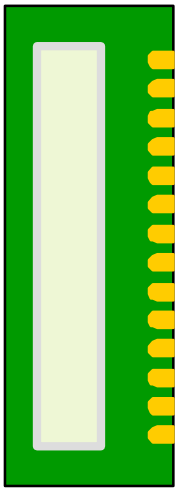
Localización en display virtual (x,y).

x: posición horizontal (de 1 a 40).

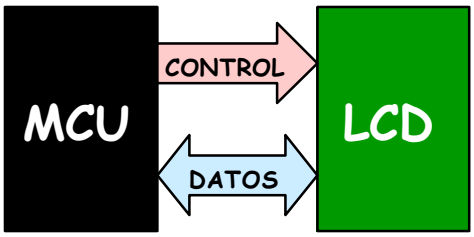
y: línea (1 ó 2).

Interface hardware LCD con driver HD44780

Pines externos.



1.-	VSS	Masa	2,7V a 5,5V
2.-	VDD	Alimentación	
3.-	VEE	Ajuste de Contraste	⬅ Máximo contraste a VSS
4.-	RS	Selección de Registro	Bits de control (entradas al driver)
5.-	R/$\overline{W}$	Lectura / Escritura	
6.-	E	Enable	
7.-	D0	Bit de Datos menos significativo	Bits de datos (entradas/salidas)
8.-	D1	Bit de Datos	
9.-	D2	Bit de Datos	
10.-	D3	Bit de Datos	
11.-	D4	Bit de Datos	
12.-	D5	Bit de Datos	
13.-	D6	Bit de Datos	
14.-	D7	Bit de Datos más significativo	



DATOS

Internos
El LCD trabaja con 8 bits

Externos
Hay dos posibilidades:
- 8 bits (D7 a D0)
- 4 bits (D7 a D4)
  - 1º los 4 bits más altos
  - 2º los 4 bits más bajos

BITS DE CONTROL

E:	Validación de datos.
R/$\overline{W}$:	Operación de lectura (1) o de escritura (0).
RS:	Selección de Registro Interno (1: datos / 0: control).

Periféricos de E/S

Control del LCD

E: Señal de validación de datos. En las transferencias de información con el LCD (lecturas o escrituras) se debe poner a 1. Si no se usa el LCD, debe permanecer a 0.

R/$\overline{W}$: Selecciona lectura (1) o escritura (0) en el LCD. Lo normal es escribir en el LCD, pero es posible leer la RAM y el estado del LCD (ocupado o disponible) y el contador de direcciones.

RS: Se selecciona uno de los dos Registros Internos del LCD.

IR (Registro de Instrucciones). Almacena códigos de instrucciones relativas al manejo del display: borrar display, desplazar cursor, definir interface a 4 u 8 bits, etc.

DR (Registro de Datos). Almacena datos a leer o escribir en RAM.

Operaciones de control en el LCD

	RS=0 (Registro de Control)	RS=1 (Registro de Datos)
R/$\overline{W}$=1	Leer flag de ocupado (BF) y puntero de direcciones (AC)	Leer contenido de DDRAM o CGRAM
R/$\overline{W}$=0	Envío de comando para funcionamiento interno	Escribir en DDRAM o CGRAM

BF: *Busy Flag* o Flag de Ocupado. Si está a 1, el LCD está en modo de operación interna y no puede procesar nuevos comandos hasta que se pone a 0.

AC: *Address Counter* o Contador de Direcciones. Es el puntero de la dirección DDRAM o CGRAM a la que se accedería con un comando de lectura o escritura a RAM. El puntero se incrementa o se decrementa (según el modo elegido) de manera automática.

Secuencias de operación

Escritura en un Registro del LCD

1. E ← 0
2. RS ← 0 ó 1 y R/$\overline{W}$ ← 0
3. E ← 1
4. Situar dato en el bus
5. E ← 0

Lectura de un Registro del LCD

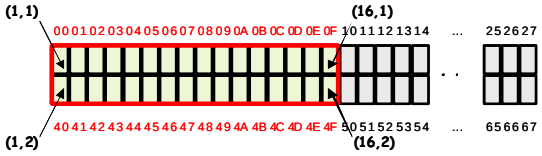
1. E ← 0
2. RS ← 0 ó 1 y R/$\overline{W}$ ← 1
3. E ← 1
4. Leer dato del bus
5. E ← 0

Descripción de los comandos

Borrar Display: **0 0 0 0 0 0 0 1**

Borra todas las posiciones de la DDRAM y sitúa el display real en la posición inicial.

Además sitúa el puntero en la posición 0 de la DDRAM.



Cursor a "Casa": **0 0 0 0 0 0 1 x**

El cursor es un indicador de la posición que se puede escribir a continuación en el LCD. Indica la posición actual del puntero de direcciones. Este comando envía el cursor a la posición (1,1) (puntero en 0x00) y el display real se sitúa en la posición inicial.

No se modifica el contenido de la DDRAM

Periféricos de E/S

Modo de Funcionamiento: **0 0 0 0 0 1 I/D S**

El bit **I/D** especifica incremento y desplazamiento del cursor a la derecha (**I/D=1**) o decremento y desplazamiento del cursor a la izquierda (**I/D=0**) tras realizar una lectura o escritura en DDRAM.

Si **S=1**, se desplaza el display real cada vez que se imprime un carácter. El desplazamiento será a la izquierda o a la derecha según el valor de **I/D**.

Control de Display, Cursor y Parpadeo: **0 0 0 0 1 D C B**

Si **D=0**, el LCD no muestra nada pero la DDRAM mantiene su contenido. Se pueden enviar y leer datos normalmente, pero no aparecerá nada en pantalla. Para volver a visualizar normalmente los caracteres de la DDRAM hay que poner **D=1**.

Si **C=1**, se hace visible el cursor que indica la siguiente posición donde se imprimiría el siguiente carácter que se envíe.

Si **B=1**, el carácter situado en la posición del cursor parpadea con una frecuencia aproximada de 2Hz.

Desplazar Cursor / Display: **0 0 0 1 S/C R/L x x**

Se emplea para desplazar una posición a la derecha (**R/L=1**) o a la izquierda (**R/L=0**) el cursor o el display real sin escribir o leer la DDRAM.

Si lo que se desplaza es el cursor (**S/C=0**), también se modifica el contador de direcciones. Si se desplaza el display real (**S/C=1**), no cambia el puntero de direcciones de la DDRAM.

Si el display se define de una línea, al llegar a la posición final (carácter 40º) se volvería a la primera con un desplazamiento del cursor. Si el display está definido para 2 líneas, tras el 40º carácter de la primera línea se pasaría al principio de la segunda línea.

Transferencia y Representación: **0 0 1 DL N F x x**

El bit **DL** define el tamaño del interface de datos externo. Si **DL=1**, será de 8 bits y si **DL=0**, será de 4 bits.

Si **N=1**, se gestionan dos líneas; si **N=0**, se tendrá una única línea activa en el display.

Si **F=1**, se emplean patrones de tamaño 5×10; si **F=0**, son de 5×8 puntos.

Periféricos de E/S

Situar Puntero en DDRAM: **1 A6 A5 A4 A3 A2 A1 A0**

A6-A0 válidas de 0x00 a 0x27 para la primera línea
A6-A0 válidas de 0x40 a 0x67 para la segunda línea

Leer Flag de Ocupado y Puntero de Direcciones

Con la combinación adecuada en **RS** y **R/$\overline{W}$** las líneas de datos del LCD pasan a ser salidas y en el puerto del MCU se puede leer el estado del BF (bit 7) y la dirección actual del contador (bits 6 a 0).

Enviar Datos a DDRAM

Se carga la dirección de la DDRAM a la que esté apuntando el contador de direcciones y éste se incrementa o decrementa dependiendo del estado configurado con **I/D**.

Leer Contenido de DDRAM

Se lee el contenido de la posición DDRAM a la que apunte el contador de direcciones. Tras la lectura, este contador se incrementa o decrementa dependiendo del modo configurado con **I/D**.

Procesado de los comandos

El LCD precisa de un cierto tiempo para procesar los comandos que se le van enviando. Para que se ejecute un determinado comando, es necesario que se haya finalizado el anterior.

Esto puede asegurare de dos modos:

a) Esperar a que el Flag de Ocupado (BF) pase a 0.

b) Establecer pausas entre comandos que sean superiores a los tiempos máximos especificados para cada comando.

En el encendido se produce un reset de inicialización con varios efectos preestablecidos.

Se produce tras superar los 4,5V y dura unos 10ms.

DL=1	(8 bits)	**N=0**	(1 línea)	**F=0**	(5×8 puntos)
D=0	(display off)	**C=0**	(cursor off)	**B=0**	(sin parpadeo)
I/D=1	(incremento)	**S=0**	(sin desplaz.)		

Durante todo el proceso de inicialización se tiene BF=1.

LCD en el compilador C de CCS

El compilador C de CCS incluye un driver para manejar LCDs: el fichero *lcd.c* que define las funciones indicadas a continuación.

lcd_init ();

Debe llamarse antes que ninguna otra función del fichero *lcd.c*.

Tal y como aparece en el fichero, además de borrar el display, configura el LCD para trabajar como sigue:

a) En formato de 4 bits, con dos líneas y con caracteres de 5×8 puntos.

b) Con display encendido, cursor apagado y sin parpadeo.

c) Con autoincremento del puntero de direcciones y sin desplazamiento del display real.

lcd_gotoxy (**x** , **y**);

Establece la posición del LCD a la que se debe acceder.

Recuérdese que la primera posición de la primera línea tiene coordenadas (1 , 1), y que la primera posición de la segunda línea es la (1 , 2).

lcd_putc (**dato**);

Escribe **dato** en la posición a la que apunta el puntero de direcciones.

La variable **dato** es de tipo *char*, y se definen algunos caracteres especiales:

\f	Borra el display
\n	Se posiciona en el inicio de la segunda línea
\b	Retrocede una posición

lcd_getc (**x , y**);

Devuelve el carácter que ocupa la posición **(x , y)** del LCD.

Por defecto, este driver usa siete bits del puerto B para establecer la comunicación entre el LCD y el microcontrolador.

B0	Enable	B4	Bit de datos D4
B1	RS	B5	Bit de datos D5
B2	R/$\overline{W}$	B6	Bit de datos D6
B3	-	B7	Bit de datos D7

También es posible escribir en el LCD usando la siguiente función.

printf(lcd_putc,**cadena**,**vars**);

cadena: Cadena de caracteres que puede formarse usando el contenido de una o más variables.

vars: Variables incluidas en la cadena (separadas por comas).

Para indicar la posición y el tipo de las variables a incluir en la cadena, se usa el formato %wt, donde w es opcional.

w:	1-9	Indica el número de caracteres a mostrar
(opcional)	01-09	Indica el número de ceros a la izquierda
	1.1-9.9	Indica cuántos decimales se han de mostrar

t:				
	c	Carácter	e	Flotante (formato exp)
	s	Cadena o carácter	f	Flotante
	u	Entero sin signo	lu	Entero largo sin signo
	x ó X	Entero hexadecimal	lx ó lX	Entero largo hexadecimal
	d	Entero con signo	ld	Entero largo con signo
	%	Simplemente un '%'		

Algunos ejemplos.

```
printf(lcd_putc , "Hola");
printf(lcd_putc , "Tecla %c pulsada %u veces" , key , cont );
```

Ejemplos de formato.

Especificador	Valor=18	Valor=254
%03u	018	254
%u	18	254
%2u	18	???
%3u	_18	254
%d	_18	-2
%x	12	fe
%X	12	FE
%4X	0012	00FE

NOTA: Sólo se puede visualizar enteros y caracteres en el LCD.
El compilador de CCS no admite salida formateada de flotantes.

La comunicación entre el microcontrolador y el LCD se lleva a cabo mediante 7 bits del Puerto B usando el driver lcd.c de CCS.

Este hecho haría que la conexión de un LCD fuera incompatible con la conexión típica de un teclado matricial 4ó4.

La inicialización que por defecto lleva a cabo este driver también puede modificarse sin más que cambiar un par de líneas del mismo.

En concreto hay que cambiar 3 ó 4 valores que se hacen llegar al LCD como comando.

El driver lcd.c está pensado para manejar un LCD usando 4 bits de datos externos al interface.

Sería posible adaptarlo para que usara 8 bits. El driver lcd_ATE.c que se facilita con este curso lleva a cabo las modificaciones necesarias.

Para evitar conflicto con la conexión de teclados matriciales, haremos que el LCD se comunique con el microcontrolador mediante el puerto D (ubicado en la dirección 08h) en lugar de hacerlo a través del puerto B.

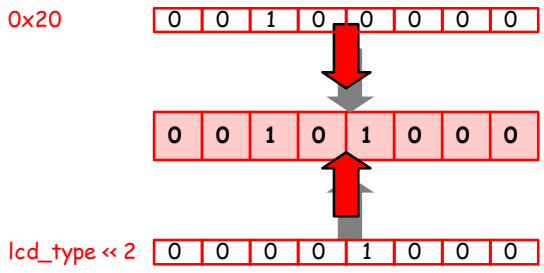
```
#if defined (__PCH__)
    #byte lcd = 0xF81          // This puts the entire structure
#else                          // on to port B (at address 6)
    #byte lcd = 6
#endif
```

```
#if defined (__PCH__)
    #byte lcd = 0xF83          // This puts the entire structure
#else                          // on to port D (at address 8)
    #byte lcd = 8
#endif
```



```
#define lcd_type 2          // 0=5x7, 1=5x10, 2=2 lines
#define lcd_line_two 0x40   // LCD RAM address for the second line

byte CONST LCD_INIT_STRING[4] = {0x20 | (lcd_type << 2), 0xc, 1, 6};
                            // These bytes need to be sent to the LCD
                            // to start it up.
```



Transferencia y Representación

0 0 1 DL N F x x

DL → Nº bits de datos
N → Nº líneas
F → Nº ptos/carácter

```
#define lcd_type 2        // 0=5x7, 1=5x10, 2=2 lines
#define lcd_line_two 0x40 // LCD RAM address for the second line

byte CONST LCD_INIT_STRING[4] = {0x20 | (lcd_type << 2), 0xc, 1, 6};
                          // These bytes need to be sent to the LCD
                          // to start it up.
```

0xc	0	0	0	0	1	1	0	0	
	0	0	0	0	0	1	1	0	6

Control del Display y del Cursor
0 0 0 0 1 D C B

D → Display on/off
C → Cursor on/off
B → Parpadeo on/off

Modo de Funcionamiento
0 0 0 0 0 1 I/D S

I/D → Increm. / Decrem.
S → Shift display on/off

El driver lcd_ATE.c lleva a cabo una comunicación con el LCD usando 8 bits de datos. En su versión original las líneas usadas son:

RA1=RS - RA2=R/W - RA3=E
RD<0:7>=D0ûD7

Como sucede con el driver lcd.c, también en este caso es posible cambiar la asignación de los pines.

```
struct lcd_pines_control {
    Bit 0 ø  boolean nada;
    Bit 1 ø  boolean rs;
    Bit 2 ø  boolean rw;
    Bit 3 ø  boolean enable;
Resto bits ø  int     otros : 4;
        } lcd_control;


struct lcd_pines_datos {
          int     datos:8;
      } lcd_datos;          ï 8 bits
                              de datos

#byte lcd_control = 5
#byte trisa = 0x85          PORTA

#byte lcd_datos = 8
#byte trisd = 0x88          PORTD
```

Es posible cambiar los bits de configuración para conseguir que el LCD se comporte como desea el usuario.

Lo único que es obligatorio respetar es el bit DL=1 (comunicación a 8 bits)

```
lcd_send_byte(0,0b00111000);     // Se envía 0 0 1 DL N F - -
lcd_send_byte(0,0b00001100);     // Se envía 0 0 0 0 1 D C B
lcd_send_byte(0,0b00000001);     // Se envía Clear Display
lcd_send_byte(0,0b00000110);     // Se envía 0 0 0 0 0 1 I/D S
```

El driver lcd_ATE.c define alguna función que no se recoge en el driver lcd.c suministrado por CCS.

```
//  lcd_putc(\t);         Avanza el cursor una posición.
//  lcd_putc(\r);         Retrocede el display real una posición.
//  lcd_putc(\v);         Avanza el display real una posición.

//  lcd_clr_line(line);   Borra la línea 'line' y sitúa el cursor
//                        en la primera posición de dicha línea.
```