DISPLAYS DE CRISTAL LIQUIDO

FUNCIONAMIENTO Y CONEXION A LA TARJETA CT6811

INDICE

PARTE 1: MANUAL DE REFERENCIA DEL LCD

PARTE II: CONEXION DEL LCD A LA TARJETA CT6811

4.- PROGRAMACION DEL LCD A TRAVES DEL CTSERVER

PARTE 1:

MANUAL DE REFENCIA DEL LCD

1.- INTRODUCCION

En esta parte se trata sobre los detalles de funcionamiento de un LCD de 2 líneas de 16 caracteres. Todos los displays de este tipo disponibles en el mercado son compatibles entre sí.

En la **seccion 2** se trata sobre las características principales del display: caracteres, memoria interna, alimentación, aspecto físico. Esta sección debe ser leída por cualquiera que vaya a manejar un LCD. **La sección 3** trata sobre los detalles relacionados con el Hardware: asignación de pines, bus de datos, de control, control del contraste, cronogramas. Finalmente en la **sección 4** se encuentran explicados los comandos de control del LCD y la secuencia de inicialización.

En esta parte se supone que el lector no conoce nada acerca de los LCD. Por ello es imprescindible para familiarizarse con su manejo y control. Si el usuario quiere saber cómo conectarlo a la tarjeta CT6811 y cómo programarlo deberá acudir a la parte II.

2.- CARACTERISTICAS DEL DISPLAY

2.1.- ASPECTO FISICO

El LCD tiene un aspecto físico como el mostrado en la figura 1. Está constituido por un circuito impreso en el que están integrados los controladores del display y los pines para la conexión del display. 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 2.

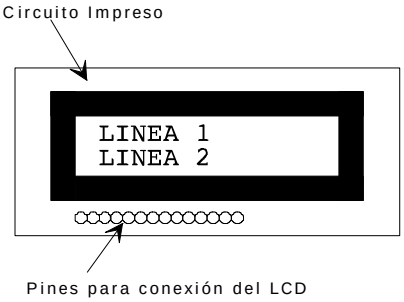


Figura 1: Aspecto físico del LCD

A pesar de que el display sólo puede visualizar 16 caracteres por línea, puede almacenar en total 40 por línea. Es el usuario el que especifica qué 16 caracteres son los que se van a visualizar.

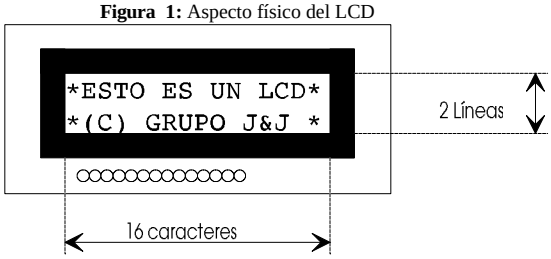


Figura 2: Capacidad de visualización de caracteres del display

2.2.- ALIMENTACION

La tensión nominal de alimentación es de 5V, con un consumo menor de 5mA.

2.3.- LOS CARACTERES DEL LCD

El LCD dispone de una matriz de 5x8 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 figura 3 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.

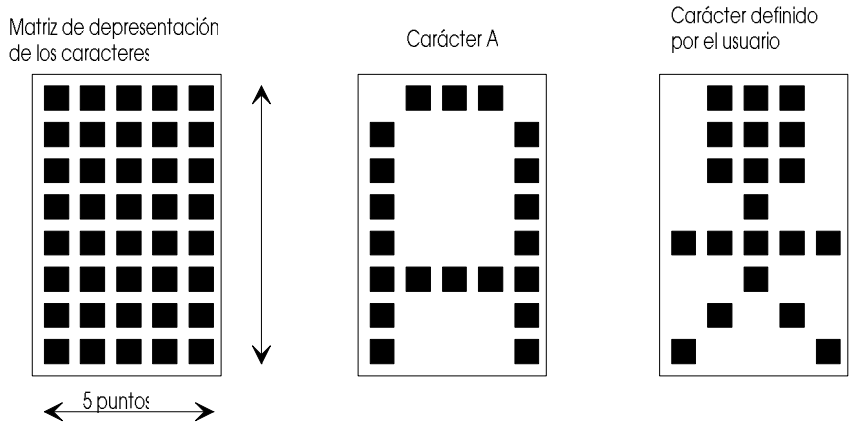


Figura 3: Matriz de representación de caracteres, representación del carácter A y de un carácter definido por el usuario

Código	Car.	Código	Car.	Código	Car.	Código	Car.	Código	Car.	Código	Car.
$20	espacio	$30	0	$40		$50	P	$60	‘	$70	p
$21	!	$31	1	$41	A	$51	Q	$61	a	$71	q
$22	“	$32	2	$42	B	$52	R	$62	b	$72	r
$23	#	$33	3	$43	C	$53	S	$63	c	$73	s
$24	$	$34	4	$44	D	$54	T	$64	d	$74	t
$25	%	$35	5	$45	E	$55	U	$65	e	$75	u
$26	&	$36	6	$46	F	$56	V	$66	f	$76	v
$27	‘	$37	7	$47	G	$57	W	$67	g	$77	w
$28	(	$38	8	$48	H	$58	X	$68	h	$78	x
$29	)	$39	9	$49	I	$59	Y	$69	i	$79	y
$2A	*	$3A	:	$4A	J	$5A	Z	$6A	j	$7A	z
$2B	+	$3B	;	$4B	K	$5B	[	$6B	k	$7B	{
$2C	,	$3C	<	$4C	L	$5C		$6C	l	$7C	|
$2D	-	$3D	=	$4D	M	$5D	]	$6D	m	$7D	}
$2E	.	$3E	>	$4E	N	$5E	^	$6E	n	$7E	→
$2F	/	$3F	?	$4F	O	$5F	_	$6F	o	$7F	←

Tabla 1: Código asociado a cada carácter imprimible por el display.

En la tabla 1 se muestran los caracteres más importantes que es capaz de imprimir el display. Todos los códigos están en hexadecimal. No se han representado los carácteres correspondientes a los códigos desde el $80 hasta el $FF, que corresponden a símbolos extraños. **Los códigos comprendidos entre el 0 y el 7 están reservados para que el usuario los defina.**

2.4.- LA MEMORIA DEL LCD

El LCD disponde de dos tipos de memorias independientes: la **DD RAM** y la **CG RAM**

2.4.1.- DD RAM (Display Data Ram)

En esta memoria se almacenan los caracteres que están siendo visualizados o que se encuentran en posiciones no visibles. El display almacena en esta memoria dos líneas de 40 caracteres pero sólo se visualizan 2 líneas de 16 caracteres. Por ello la **DD RAM** tiene un tamaño de 2x40=80 bytes.

Debido a esta pecualiar disposición de la **DD RAM** se puede pensar en el display como un *display virtual* constituido por dos líneas de 40 caracteres cada una (Fig. 4). La posición situada más a la izquierda de cada línea es la **posición 1** y la situada más a la derecha es la **posición 40.** Para localizar los elementos dentro del *display virtual* se va a utilizar un par de coordenadas (x,y) donde x representa la posición horizontal (comprendida entre 1-40) e y representa la línea (1-2). El display real es una *ventana* en la que se visualizan dos líneas de 16 caracteres. Es lo que el usuario está viendo. En el ejemplo de la figura 4 se encuentra almacenado en la línea 1 del display virtual el mensaje: “ESTO ES UNA PRUEBA DE UN MENSAJE”. Sin embargo, en este ejemplo, el usuario sólo verá en el display el mensaje “PRUEBA DE UN MEN” que tiene exactamente 16 caracteres de longitud. Más adelante se verá cómo es posible ‘mover’ el display real para que se desplace a lo largo del display virtual. Tal y como se encuentra configurado el display real en la figura 4, la posición (14,1) se corresponde con la letra P, la posición (15,1) con la letra R, ..., y la posición (29,1) con la letra N. Cuando se inicializa el LCD, el display real se sitúa en el extremo más izquierdo del display virtual, que se corresponde con los valores de x comprendidos entre 1 y 16. En la figura 5 se muestra la situación del display real respecto al virtual al inicializar el LCD.

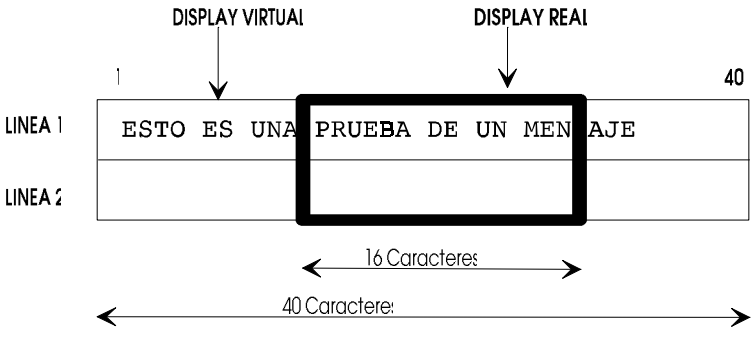


Figura 4: Display virtual y display real

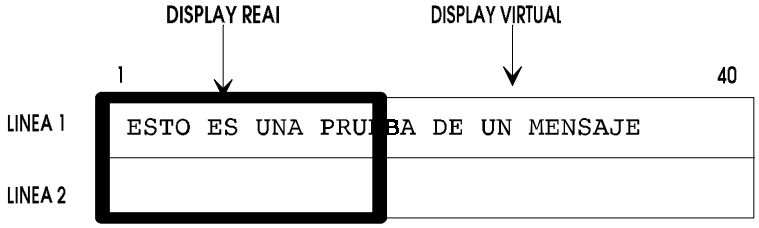


Figura 5: Posición del display real respecto al virtual cuando se inicializa el LCD

En el ejemplo de la figura 5, en la posición (2,1) se encuentra la letra E y en la posición (16,1) la letra U.

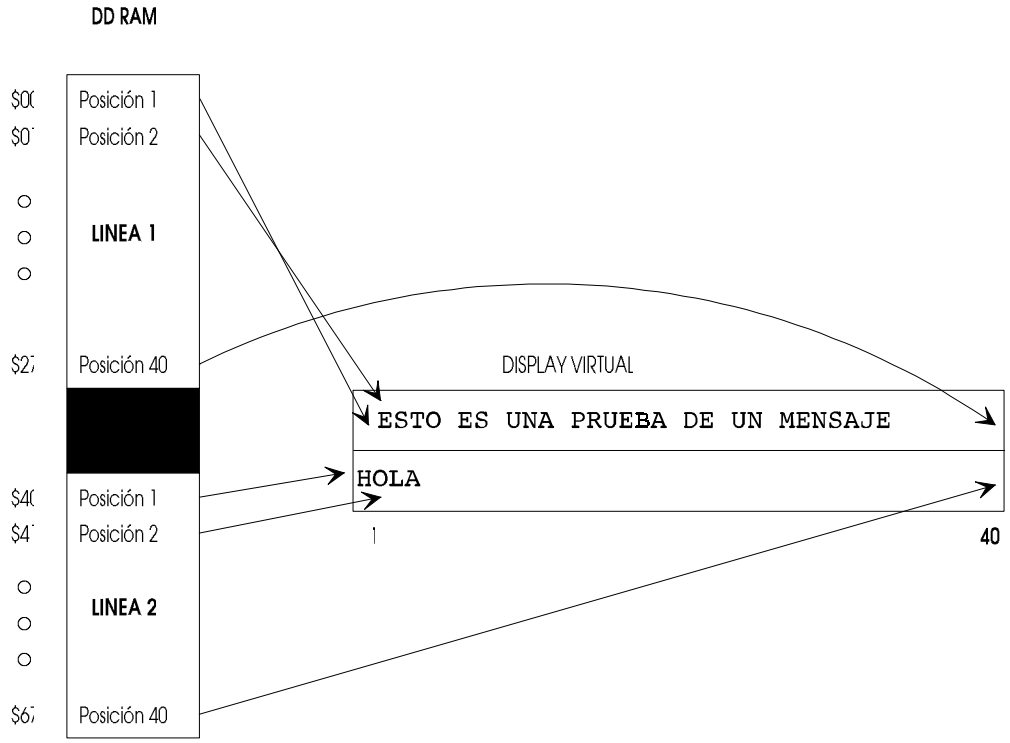


Figura 6: Mapa de memoria de la DD RAM y sus posiciones en el display virtual.

El mapa de memoria de la **DD RAM** se muestra en la figura 6. Está constituido por dos bloques de 40 bytes. El primer bloque se corresponde con los 40 caracteres de la línea 1 del display virtual. El segundo bloque con la segunda línea. En la figura se han representado las direcciones en hexadecimal. Así, las direcciones $00-$27 están asociadas a las posiciones (1,1)-(40,1) del display virtual y las direcciones $40-$67 a las posiciones (1,2)-(40,2). En el ejemplo, en la dirección $40 de la DD RAM se encuentra almacenado el carácter H, que se corresponde con la posición (1,2) del display virtual. En la dirección $02 se encuentra el carácter S, posición (3,1) del display virtual. Nótese que **los bloques de memoria asociados a la línea 1 y 2 no son contiguos.**

Las operaciones de escritura en el display, en realidad son operaciones de escritura en la memoria DD RAM. Según en la posición de esta memoria en la que se escriba el carácter, aparecerá en una posición u otra en el display real. Los caracteres enviados al display pueden ser visibles si se encuentran en posiciones que caen dentro del display real o pueden ser no visibles. En la figura 5, las posiciones (1,1)-(16,1) y (1,2)-(16,2) son visibles. Todas los caracteres enviados a esas posicones serán visibles. Si se envía un carácter a cualquiera de las otras posiciones no será visible.

2.4.2.- LA CG RAM (Character Generator RAM)

La CG RAM es la memoria que contiene los caracteres definibles por el usuario. Está formada por 64 posiciones, con direcciones $00-$3F. Cada posición es de 5 bits.

La memoria está dividida en 8 bloques, correspondiendo cada bloque a un carácter definible por el usuario. Por ello el usuario puede definir como máximo 8 caracteres, cuyos códigos van del 0 al 7. En la figura 7 se ha representado la CG RAM. Todas las direcciones están en hexadecimal.

Como se representó en la figura 3, cada carácter está constituido por una matriz de 5 columnas x 8 filas. Para definir un carácter y asignarle por ejemplo el código 0 habrá que almacenar en las posiciones $00-$07 los valores binarios de las 8 filas del carácter del usuario. Un bit con valor 1 representa un punto encendido. Un bit con valor 0 es un punto apagado.

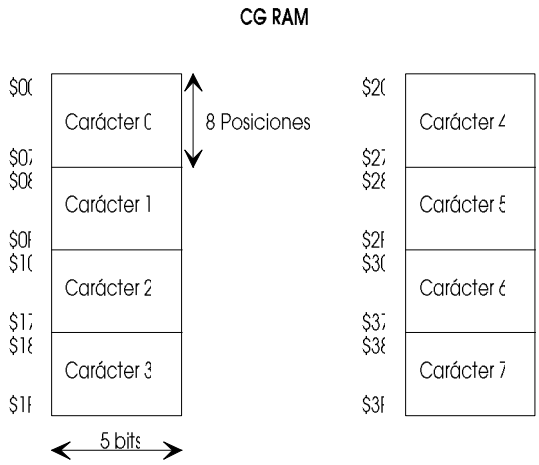


Figura 7: Mapa de memoria de la CG RAM

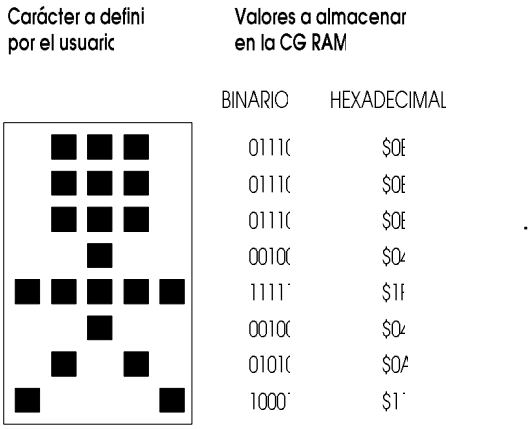


Figura 8: Caracter definido por el usuario y los valores a almacenar en la CG RAM

En la figura 8 se ha dibujado un carácter que se quiere definir. A la Derecha del dibujo se encuentran los valores en binario y en hexadecimal que hay que almacenar en las posiciones de la CG RAM. Si se quiere que este carácter tenga asignado el código 0 habrá que almacenar el valor $0E en la posición $00, $01 y $02, el valor $04 en la $03.... y el valor $11 en la posición $07, como se muestra en la figura 9.

Una vez definido el nuevo carácter, cada vez que se envie su código correspondiente al display se visualizará.

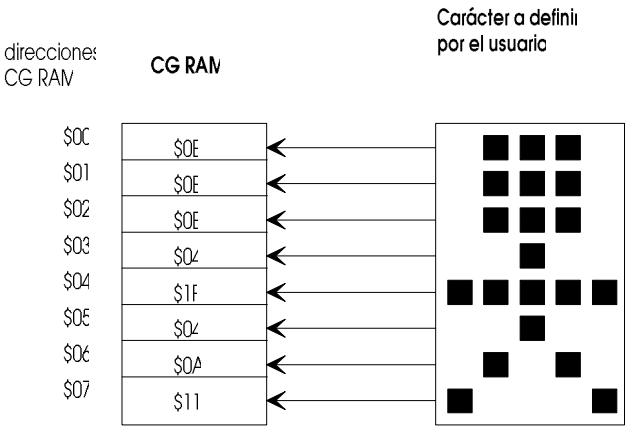
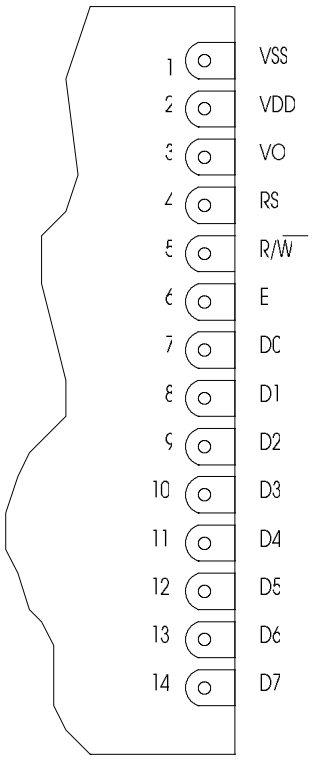


Figura 9: Valores a almacenar en la CG RAM para definir el carácter 0

3.- INTERFAZ HARDWARE

3.1.- Asignación de pines



Nº de PIN	Simbolo	Descripción
1	VSS	Masc
2	VDD	Alimentación
3	VC	Voltaje de ajuste del contraste
4	RS	Selección de registro
5	R/W	Lectura/escritura
6	E	Enable
7	DC	Bit de datos menos significativo
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 mas significativo

Figura 10: Asignación de pines del LCD

3.2.- El interfaz del display con el mundo exterior

En la figura 11 aparecen las señales necesarias para el funcionamiento y control del display. Los datos se transmiten por un bus de datos de 8 bits de anchura (El display ofrece la posibilidad de trabajar con este bus multiplexado en dos grupos de 4 bits, pero esto se verá más adelante). Para el control del display son necesarios 3 bits: una señal de ***enable*** **(E),** una para indicar **lectura/escritura (R/W)** y otra para seleccionar uno de los dos registros internos **(RS).** Por ello, en el caso peor, el sistema de control del display necesitará utilizar 8+3=11 bits. En el caso de utilizarse la tarjeta CT6811 será necesario utilizar 2 puertos, por ejemplo el PUERTO C para los datos y el PUERTO B para el control.

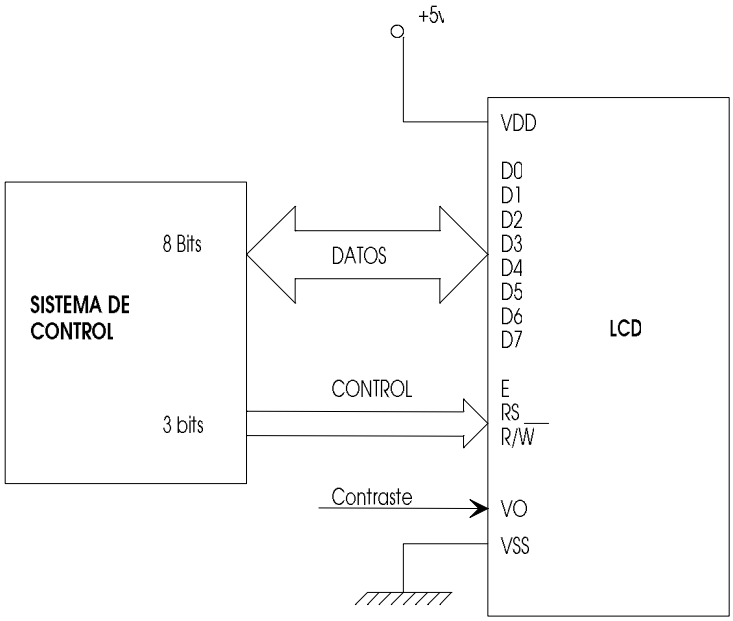


Figura 11: Interfaz del LCD con un sistema de control

3.3.- El bus de datos

El bus de datos del display se puede configurar para funcionar de dos formas diferentes. Bien como un bus de 8 bits o bien como un bus multiplexado de 4 bits. El utilizar el bus multiplexado de 4 bits es una opción muy útil para ahorrar bits en el sistema de control. En vez de utilizar 11 bits en total, se utilizan 7. (En el caso de la CT6811 sólo con el PUERTO C se podria controlar el display entero). Se ahorran bits pero se gana en complejidad del controlador, que tiene que multiplexar y demultiplexar los datos. Al utilizar un bus de 8 bits hacemos que el controlador sea más sencillo pero se 'gastan' muchos mas bits.

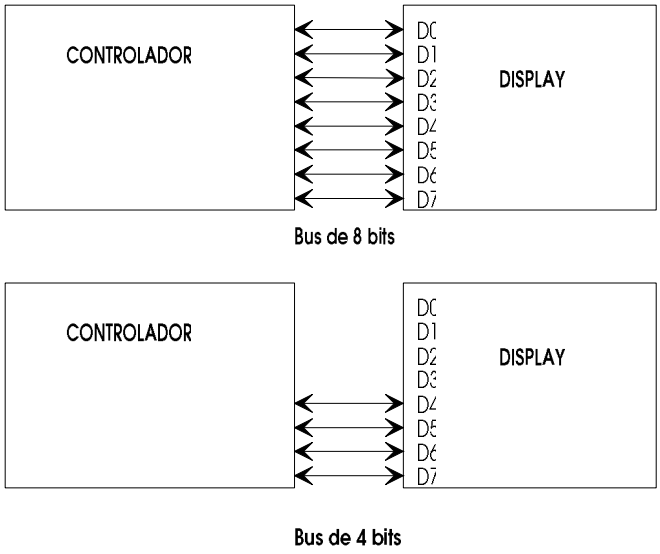


Figura 12: Conexión del LCD utilizando un bus de 8 bits y de 4 bits

En la figura 11 aparecen representados los dos tipos de buses. Cuando se utiliza un bus de 4 bits **sólo se utilizan los pines D4-D7 del display** dejándose D0-D3 'al aire'. La transferencia de la información se realiza de la siguiente manera: **primero los 4 bits más significativos y luego los 4 menos significativos**.

3.4.- El bus de control

El bus de control está formado por 3 señales: ***RS, R/W y E***. La señal **E** es la señal de validación de los datos. Cuando no se utiliza el display esta señal debe permanecer a 0. Sólo en las transferencias de información (lecturas o escrituras) es cuando se pone a nivel 1 para validar los datos, pasando despues de un tiempo a nivel 0. En la siguiente sección se explican detalladamente las temporizaciones.

La señal **R/W** permite seleccionar si la operación que se va a realizar sobre el display es una lectura o una escritura. Cuando **R/W=1** se realizan lecturas y cuando **R/W=0** escrituras. Lo normal siempre es realizar escrituras, no obstante, el display ofrece la posibilidad de poder leer los contenidos de la memoria CG RAM y DD RAM así como leer el estado interno del display (ocupado o disponible) y el contador de direcciones..

Con **RS** (Register Select) se selecciona el registro interno del display sobre el que se va a leer/escribir. El LCD dispone de dos registros internos: **Registro de control** y **registro de datos**. Ambos registros son de lectura y escritura. **RS=0** selecciona el registro de control. **RS=1** el registro de datos.

	REGISTRO DE CONTROL	REGISTRO DE DATOS
LECTURA	Lectura del flag de ocupado (D7) y del contador de direcciones (D0-D6)	Leer contenido de la memoria CG RAM o DD RAM
ESCRITURA	Ejecución de un comando interno: borrar display, desplazar el display, mover cursor...	Escribir en la DD RAM o CG RAM

En la sección 4 se tratan con detalle los diferentes comandos internos del display.

3.5.- El control del contraste

Para controlar el constraste hay que introducir por el pin Vo una tensión entre 5 y 0 voltios. La tensión típica es de 0.6 voltios. Normalmente se coloca un potenciómetro para poder ajustar en cada

momento el contraste más adecuado. En la figura 13 se muestra un esquema típico de control del contraste.

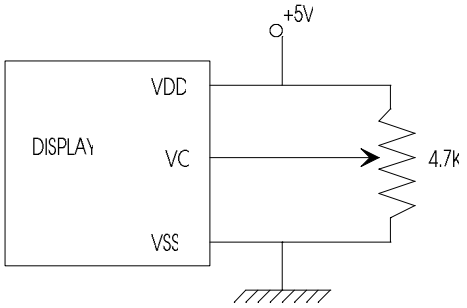


Figura 13: Control del contraste en el LCD

3.5.- Temporización

En la figura 14 se han representado los cronogramas correspondientes a una operación de escritura y otra de lectura. Al controlar el display los tiempos empleados deben ser siempre mayores que los mínimos indicados en la figura y menores que los máximos. A pesar de la aparente complejidad del cronograma, las operaciones de lectura y escritura son muy sencillas. En la figura 14 se ha supuesto que las transmisiones de los datos se realizaban a 8 bits.

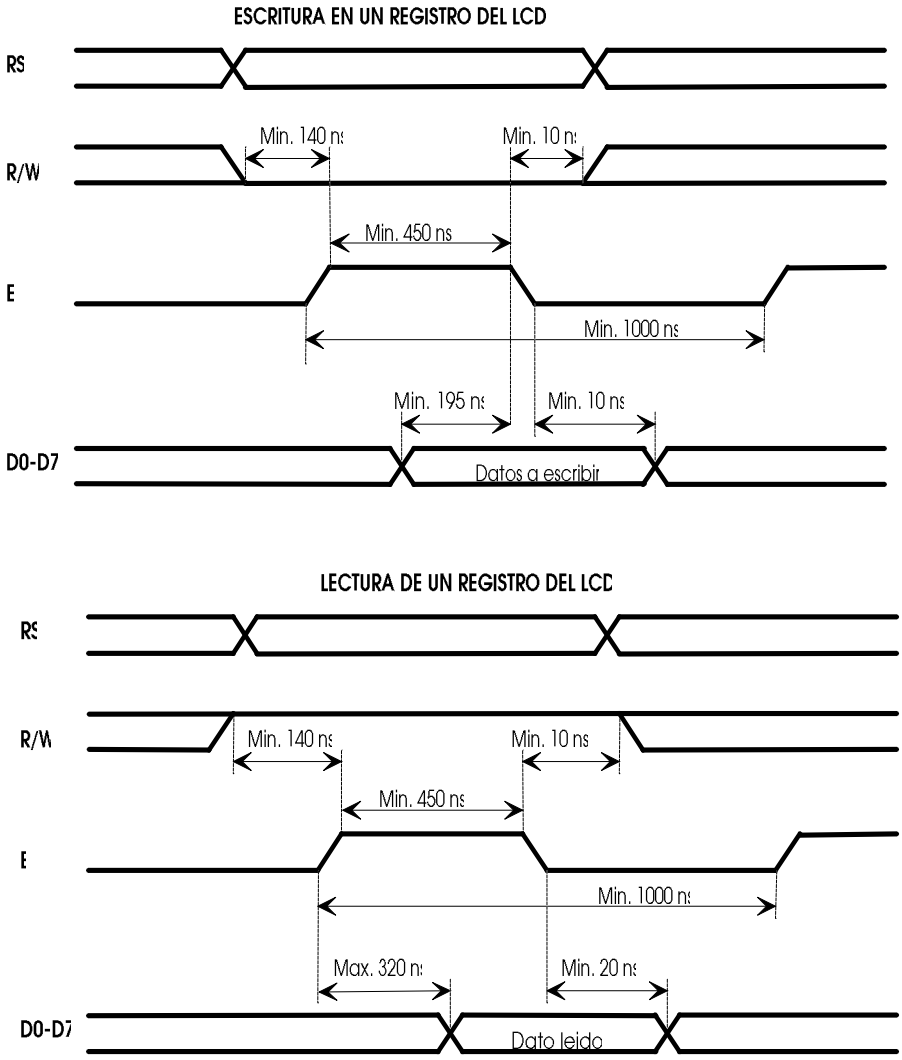


Figura 14: Cronogramas del LCD

En la figura 15 se muestran los cronogramas correspondientes a las operaciones más habituales: escritura de un carácter en el LCD. Se presenta el cronograma cuando se usa un bus de 8 bits y cuando se usa un bus multiplexado de 4 bits. Se ha supuesto que el carácter enviado es el $41 (La letra 'A').7

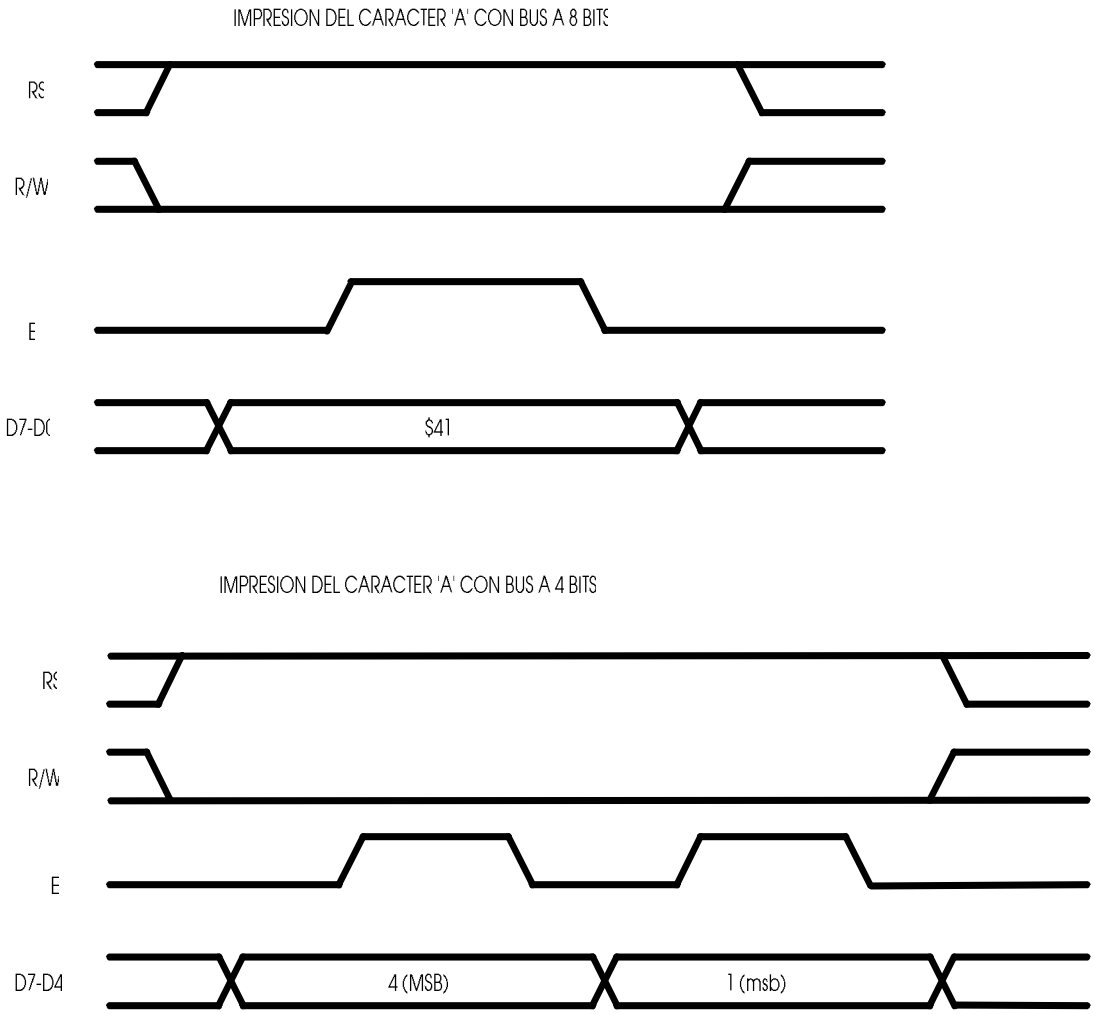


Figura 15: Cronogramas correspondientes a la operación de escribir el dato $41 (Carácter 'A') en el display utilizando un bus de datos de 8 y 4 bits.

Para el caso de 4 bits, primero se envían los 4 bits MAS SIGNIFICATIVOS y despues los 4 bits menos significativos. Los cronogramas de arriba se pueden expresar 'secuencialmente', describiendo las operaciones que tendría que realizar un microcontrolador para escribir el carácter A en el LCD

- **Operaciones a realizar para el caso de 8 bits.**

1. La señal E se encuentra siempre a 0 antes de realizar cualquier operación
2. Poner RS=1 y R/W=0
3. Situar el dato a imprimir en el bus de datos del LCD (En este ejemplo se enviaría $41)
4. E=1
5. E=0
6. El carácter ha sido imprimido en el LCD.

- **Operaciones a realizar para el caso de 4 bits:**

1. Poner RS=1 y R/W=0
2. Situar el valor 4 en el bus de datos del LCD (4 bits más significativos)

3. E=1
4. E=0
5. Situar el valor 1 en el bus de datos del LCD (4 bits menos significativos)
6. E=1
7. E=0
8. El carácter ha sido imprimido en el LCD

4.- COMANDOS DEL LCD

4.1.- Introducción

El LCD se controla mediante comandos que se envían al registro de control del LCD, seleccionado al poner la señal RS a nivel bajo (0). Cuando lo que se quiere es imprimir caracteres en el display o enviar información a la CG RAM para definir caracteres se selecciona el registro de datos poniendo RS a nivel alto (1).

Existe un *contador de direcciones* para la DD RAM y otro para la CG RAM, el cual contiene la dirección a la que se va a acceder. Modificando el *contador de direcciones* es posible acceder a cualquier posición tanto de la CG RAM como de la DD RAM. Con ello se consigue por ejemplo imprimir caracteres en cualquier posición del LCD. Cada vez que se realiza un acceso a memoria, el contador de direcciones se incrementa o decrementa automáticamente, según cómo se haya configurado el LCD.

Al LCD le lleva un cierto tiempo procesar cada comando enviado. Por ello, para que se ejecute el comando especificado es necesario asegurarse de que el comando anterior ha finalizado. Existen dos estrategias para realizar esto. La primera se basa en leer del display el **bit de ocupado**. Si este bit se encuentra a 1 quiere decir que el LCD está ocupado procesando el comando anterior y por tanto no puede procesar nuevos comandos. La segunda estrategia, menos elegante pero más cómoda de implementar, consiste en realizar una pausa antes de volver a enviar el siguiente comando. Los tiempos máximos que tarda el display en procesar los comandos están especificados por el fabricante y tienen un valor típico de 40µs. Si se realiza una pausa mayor o igual a esta se tiene garantía de que el display ha terminado de ejecutar el comando.

4.2.- Resumen de comandos

	RS	R/W	D7	D6	D5	D4	D3	D2	D1	D0
Borrar Display	0	0	0	0	0	0	0	0	0	1
Cursor a Home	0	0	0	0	0	0	0	0	1	*
Establecer modo de funcionamiento	0	0	0	0	0	0	0	1	I/D	S
Control ON/OFF	0	0	0	0	0	0	1	D	C	B
Desplazamiento del cursos/display	0	0	0	0	0	1	S/C	R/L	*	*
Modo de transferencia	0	0	0	0	1	DL	1	0	*	*
Acceso a memoria CG RAM	0	0	0	1	Dirección de la CG RAM					
Acceso a memoria DD RAM	0	0	1	Dirección de la DD RAM						
Lectura de dirección y del flag de ocupado	0	1	BF	Contador de dirección						
Escritura de datos en CG RAM /DD RAM	1	0	Dato a escribir							
Lectura de datos en CG RAM/DD RAM	1	1	Dato leido							

I/D = 1: Incrementar contador direcciones
S=1: Desplazamiento del display
D=1: Display ON
C=1: Cursor ON
B=1: Parpadeo del caracter en la posición del cursor
S/C=1: Desplazar el display.
R/L=1: Desplazamiento a la derecha
DL=1: Configurar display a 8 bits
BF=1: Display ocupado

I/D = 0: Decrementar contador direccione
S=0: Display "quieto
D=0: Display OFI
C=2: Cursor OFI
B=0: No hay parpadec
S/C=0: Desplazar el curso
R/L=0: Desplazamiento a la izquierdc
DL=0: Configurar display a 4 bit
BF=0: Display listo para ejecutar otra operació

Figura 16: Resumen de los comandos del LCD

4.3.- Descripción de los comandos

4.3.1.- Borrar el display

- **DESCRIPCION:** Este comando borra ***todas las posiones*** del display virtual y sitúa el display real en la posición inicial (Figura 5), en la que se visualizan las posiciones desde la (1,1) hasta la (16,1) y desde la (1,2) hasta la (16,2). El cursor se sitúa en la posición (1,1) (Dirección 0 de la DD RAM).
- **CODIGO:** $01
- **TIEMPO DE EJECUCION**: 1.64ms

4.3.2.- Cursor a HOME

- **DESCRIPCION:** Enviar el cursor a la posición (1,1). El display real se sitúa en la posición inicial. (Figura 5).
- **CODIGO:** $02
- **TIEMPO DE EJECUCION: 1.64ms**

4.3.3.- Establecer modo de funcionamiento

- **CODIGO:** 0 0 0 0 0 1 I/D S
- **DESCRIPCION**: Actualizar el contador de direcciones en la forma especificada y establecer si el display realiza desplazamientos o no. Estas acciones se llevan a cabo cada vez que se realiza una lectura o escritura en el display. Cuando I/D=1, el *contador de direcciones* se incrementa, lo que provoca que el cursor avance hacia la derecha cada vez que se imprime un carácter en el display. Cuando I/D=0 el *contador* se decrementa y el cursor se mueve hacia la izquierda al imprimir. Con S=1 se indica al LCD que debe mover el display real una posición a la derecha cada vez que se imprime un carácter. Con S=0 el display debe permanecer 'quieto' al imprimir. (Normalmente se utiliza I/D=1 y S=0, por lo que el **comando típico es 0x06**)
- **TIEMPO DE EJECUCION**: 40µ

4.3.4.- Control ON/OFF

- **CODIGO:** 0 0 0 0 1 D C B
- **CODIGO TIPICO**: $0E
- **DESCRIPCION:** Activar o desactivar el display, el cursor y el parpadeo
  1. ***Display ON/OFF***: D=1 activar el LCD. Cuando D=0 el LCD funciona normalmente pero no se visualiza ninguna información. Es posible realizar impresiones, enviar comandos, pero nada quedará reflejado en pantalla. Sólo cuando D=1 se puede ver algo en el display.
  2. ***Cursor ON/OFF***: C=1 activa el cursor. Con C=0 el cursor no se ve.
  3. ***Parpadeo ON/OFF***: B=1 hace que los caracteres situados en la posición del cursor parpadeen. Con B=0 no hay parpadeo.
- **TIEMPO DE EJECUCION:** 40µ

4.3.5.- Desplazamiento del cursor/display

- **CODIGO:** 0 0 0 1 S/C R/L 0 0
- **DESCRIPCION:** Desplazar una posición el cursor o el diplay real. Con S/C=1 se mueve el display, con S/C=0 el cursor. R/L=1 desplaza a la derecha y R/L=0 a la izquierda.
- **TIEMPO DE EJECUCION**: 40µ

4.3.6.- Modo de transferencia de la información

- **CODIGO:** 0 0 1 DL 1 0 0 0
- **DESCRIPCION**: Seleccionar el bus de datos del display para trabajar a 8 bits (DL=1) o a 4 bits (DL=0)
- **TIEMPO DE EJECUCION**: 40 µ

4.3.7.- Acceso a posiciones concretas de la CG RAM

- **CODIGO:** 0 1 A5 A4 A3 A2 A1 A0
- **CODIGO TIPICO**: 0 1 0 0 0 0 0 0 (Acceso a la posición 0 de la CG RAM)
- **DESCRIPCION:** Acceder a la dirección A5 A4 A3 A2 A1 A0 de la CG RAM. Esta es la dirección que se copia en el *contador de direcciones* de la CG RAM. La siguiente escritura en el registro de datos del display (RS=1) se copiará en la posición indicada de la CG RAM
- **TIEMPO DE EJECUCION**: 40 µ

4.3.8.- Acceso a posiciones concretas de la DD RAM

- **CODIGO:** 1 A6 A5 A4 A3 A2 A1 A0
- **CODIGO TIPICO**: 1 0 0 0 0 0 0 0 (Acceso a la posición 0 de la DD RAM)
- **DESCRIPCION**: La dirección A6 A5 A4 A3 A2 A1 A0 se copia en el *contador de direcciones* de la DD RAM. La siguiente escritura en el registro de datos (RS=1) se grabará en la posición indicada de la DD RAM.
- **TIEMPO DE EJECUCION**: 40 µ

4.3.9.- Envíar datos a la CG RAM o a la DD RAM

- **TIEMPO EJECUCION:** 40µ
- **DESCRIPCION**: Enviar un dato a la DD RAM o a la CG RAM. Por defecto se accede a la DD RAM, con lo que se imprimen los caracteres especificados en el display. La selección de una u otra memoria se realiza mediante los comandos descritos en los apartados 4.3.7 y 4.3.8. A la CG RAM se accede para definir caracteres especiales. Lo normal es acceder siempre a la DD RAM, porque es donde se va a realizar la impresión de caracteres en el display.

4.4.- SECUENCIA TIPICA DE INICIALIZACION DEL LCD

En la figura 17 se ha representado en un diagrama la secuencia de inicialización del LCD para trabajar con un bus de datos de 8 ó 4 bits. Para el caso de 8 bits no hay ningún problema, sin embargo el caso de 4 bits es un poco más complejo. Después de encender el LCD aparecerá la linea superior un poco más oscura que la inferir. Esto quiere decir que el display no ha sido inicializado todavía. En el caso de 4 bits sólo se conectan 4 bits mas significativos del LCD, dejando los otros 4 al ‘aire’. Al enviar el codigo 2 (Bits 0 0 1 0) el display se configura para trabajar a 4 bits. Se puede observar cómo la linea superior deja de estar más oscura que la inferior. A partir de este momento las transferencias hay que realizarlas en dos partes: primero se envían los 4 bits mas significativos y después los 4 bits menos significativos. Para confirmar que la transferencia es a 4 bits hay que enviar el código $28; primero los bits 0 0 1 0 y después los bits 1 0 0 0. De aquí en adelante la inicialización es igual tanto para 8 bits como para 4, con la salvedad de que en el segundo caso hay que enviar los datos multiplexados.

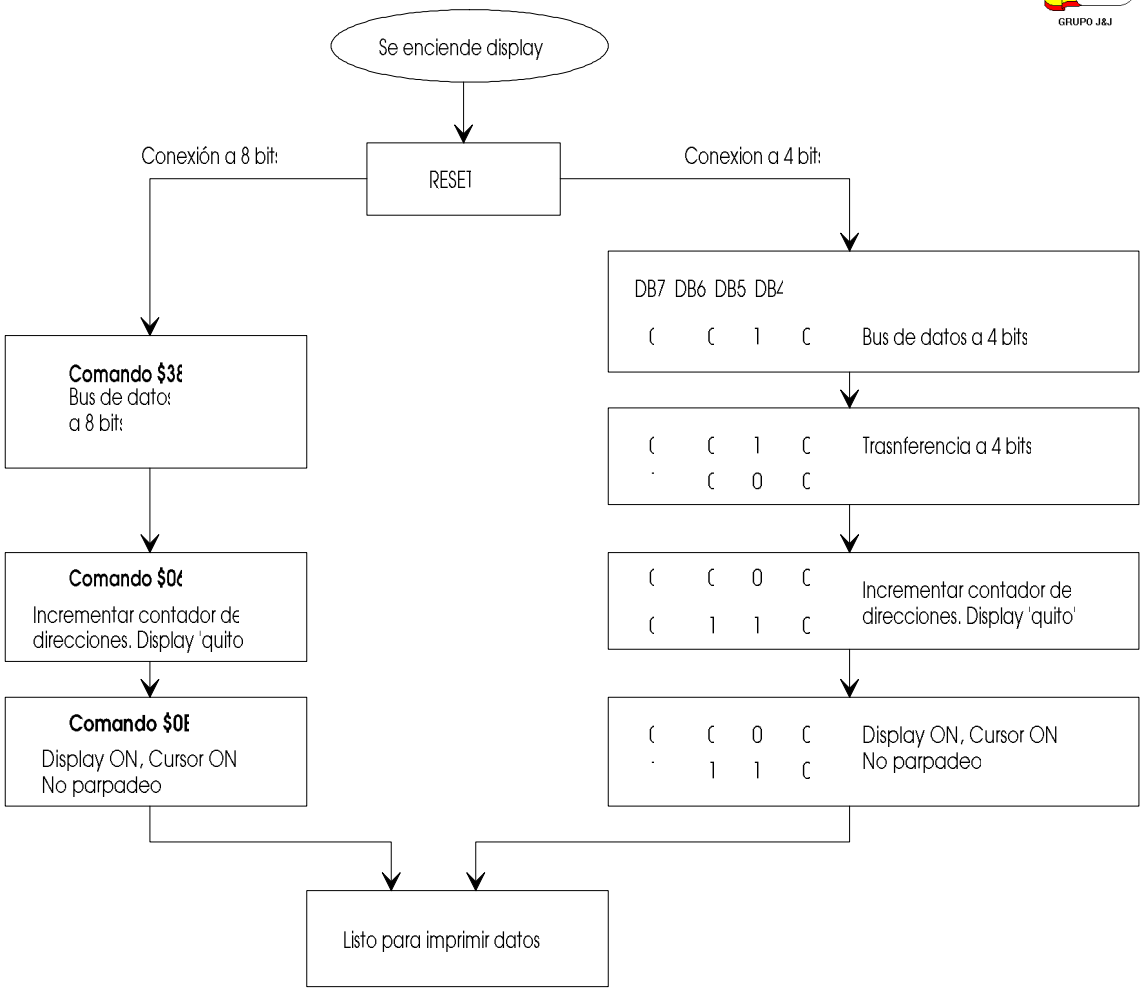


Figura 17: Códigos a enviar para inicializar el display, tanto a 8 bits como a 4.

PARTE II:

CONEXION DEL LCD A LA TARJETA CT6811

1.- INTRODUCCION

En esta segunda parte se trabaja con el LCD de una forma más concreta. El objetivo es conectarlo a la CT6811 y poderlo programar con ella. En la **sección 2** se discute sobre las distintas formas que existen de conectar el LCD a la CT6811 **.** Según la forma escogida el software diferirá. En la **sección 3** presentan programas concretos para programar el LCD. Se incluye el software para los distintos tipos de conexionado del LCD y programas de ejemplos de impresión de cadenas y de definición de caracteres. Finalmente en la **sección 4** se presenta un ejemplo de programación del LCD a través del CTSERVER.

2.- FORMAS DE CONEXION DEL LCD A LA CT6811

2.1.- Introducción

Existen muchas formas de conexión del LCD a la CT6811 según el criterio que queramos emplear. Podemos aplicar un criterio de conseguir la mayor simplicidad software, otro de minimizar los bits necesarios para su control...

Existen 3 aspectos fundamentales que determinan la forma de conexión del LCD. Son: memoria, número de bits necesarios y tipo de puertos (puertos de entrada, salida o bidireccionales). Con el conexionado elegido se pretende hacer mínimo alguno de estos factores. Por ejemplo si se ha ampliado la memoria de la tarjeta CT6811, los puertos B y C se pierden, por lo que el número de bits de salida disponibles en el 68HC11 dismunuye 9 (6 bits del puerto D y 3 del puerto A).

A lo largo de la sección 2 se pretende mostrar los distintos tipos de conexionados posibles y las ventajas y desventajas de unos y otros.

2.2- Ventajas e inconvenientes del conexionado según el tamaño del bus de datos del LCD

El bus de datos del LCD es de 8 bits, pero se puede configurar para trabajar con un bus de 4 bits, en el que se transmiten los bytes multiplexados en grupos de 4 bits. Esto nos permite ahorrar puertos del 68HC11 pero hace que el software de control sea un poco más complejo y ocupe más memoria. A continuación se muestran las características de cada tipo de conexión:

- **BUS DE DATOS A 8 bits:** En total se necesitan 8 pines de entrada/salida para los datos y 3 pines de salida para el control. ***Es necesario disponer de 8+3=11 pines,*** de los cuales 8 deben ser configurables para E/S.
    - **VENTAJAS:**
        - El software es muy sencillo
        - El software ocupa poca memoria
    - **INCONVENIENTES:**
        - Son necesarios 11 pines libres en el 68HC11
        - 8 de los pines se deben poder configurar para E/S

- **BUS DE DATOS A 4 BITS:** Se necesitan 4 pines de entrada/salida y 3 pines de salida***. Hay que disponer de 4+3=7 pines,*** de los cuales 4 deben ser configurables para E/S.
    - **VENTAJAS:**
        - Hay un ahorro de 4 pines. Se necesitan 7 pines en total.
    - **INCONVENIENTES:**
        - El software es más complejo y ocupa más espacio
        - Se necesitan 4 pines configurables para E/S

2.3.- Ventajas e inconvenientes del conexionado según el tipo de control sobre el LCD

El LCD se puede controlar de dos formas diferentes:

- **Control en bucle cerrado:** Mediante este control se leen datos lel LCD para saber si está ocupado o está listo para ejecutar otro comando que se le mande. Dentro del LCD existe un ***bit de busy*** (bit de ocupado) que indica si el LCD está realizando algún tipo de operación o no. Antes de enviar un comando al LCD habrá que comprobar el ***bit de busy***. Si está a 1 el LCD está ocupado procesando la instrucción anterior y no podrá ejecutar una nueva. Habrá que esperar a que se ponga a 0 para poder enviar el siguiente comando.
- **Control en bucle abierto:** En este tipo de control no se lee ningún dato del LCD. El fabricante especifica los tiempos máximos que tarda cada comando del LCD en ejecutarse. Después de enviar un comando habrá que realizar una espera que nos asegure que el LCD está listo para recibir nuevos comandos.

Estos dos tipos de control tienen una repercusión muy importante en el conexionado del LCD.

- **Control en bucle cerrado**
  - **VENTAJAS:**
    - El control es 'como Dios manda'.
    - No hace falta realizar ningún tipo de temporización.
  - **INCONVENIENTES:**
    - Se necesitan en la CT6811 pines configurables para E/S
    - Por cada comando enviado al LCD hay que realizar muchas operaciones software: Configurar los pines de datos para entrada, leer bit de busy, esperar a que se ponga a 0, configurar pines para salida y enviar comando.

- **Control en bucle abierto**
  - **VENTAJAS:**
    - No se necesitan pines de E/S. Sólo son necesarios pines de Salida, lo cual es muy importante porque en la CT6811 los únicos puertos de E/S son el PUERTO C y el PUERTO D.
    - No es necesario el bit de control R/W puesto que siempre se van a realizar escrituras en el LCD. Hay un ahorro de 1 bit.
    - La operación de enviar comandos es sencilla: Se envía un comando y se realiza una pausa.
  - **INCONVENIENTES:**
    - El control es 'menos serio'
    - No se puede leeer nada del LCD. Pero esto realmente no es un inconveniente porque a parte del ***bit de busy*** no interesa leer nada más.

Se observa que las ventajas del control en bucle abierto superan con creces los inconvenientes. No obstante, en las siguientes secciones se van a mostrar el software para ambos tipos de control.

2.4.- Distintas opciones de conexionados

En la siguiente tabla se muestran las distinas opciones y los pines necesarios para el control del LCD.

	Bus a 4 bits	**Bus a 8 bits**
Bucle cerrado	**OPCION A**: Necesarios 7 pines. 4 Pines debe ser de E/S	**OPCION B:** Necesarios 11 pines 4 Pines debe ser de E/S
Bucle Abierto	**OPCION C:** Necesarios 6 pines sólo de salida	**OPCION D:** Necesarios 10 bits sólo de salida

La opción C es la que ofrece el mayor ahorro en el número de pines. Sólo con 6 pines es posible controlar el LCD. En la figura 18 se muestran los esquemas de conexionado utilizados para las diferentes opciones. Todos los ejemplos utilizarán esos conexionados.

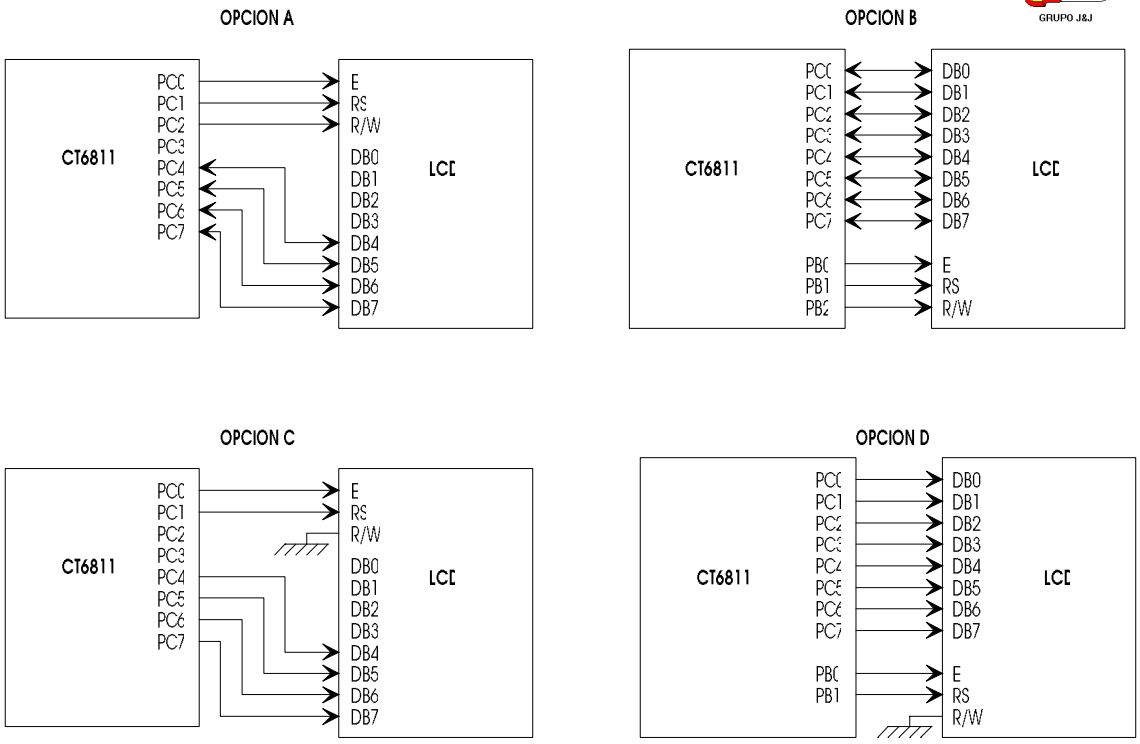


Figura 18: Conexionados empleados para las 4 opciones de utilizacion del display

3.- SOFTWARE DE CONTROL PARA EL LCD

3.1.- INTRODUCCION

En esta sección se presentan ejemplos de manejo del LCD para las 4 opciones de conexión de la figura 18. El software se encuentra dividido en dos niveles (Figura 19). En el nivel más bajo, denominado nivel Fisico, el software depende del tipo de conexionado. El interfaz que ofrece este nivel está constituido por 3 llamadas: **dato_lcd()** que permite enviar un dato a la memoria del LCD (CG RAM o DD RAM), **control_lcd()** para enviar comandos de control (CLS, Cursor a home...) e **init_lcd()** para inicializar el LCD.

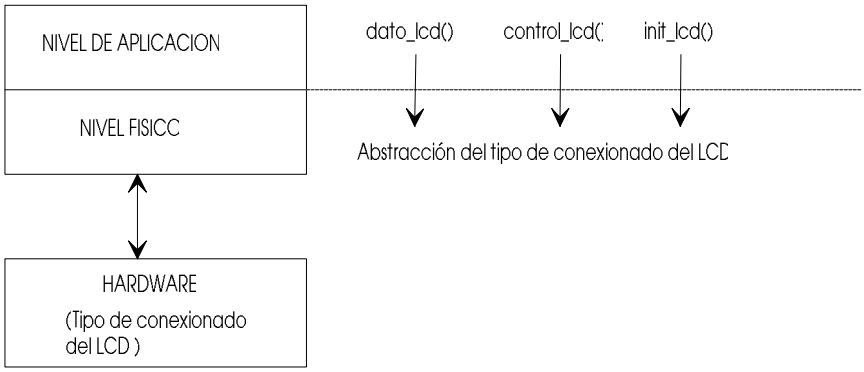


Figura 19: Niveles en los que se ha dividido el software de control

Esta disposión en dos niveles es muy cómoda. El nivel fisico depende del tipo de conexionado. Sin embargo ***el nivel de aplicación es totalmente independiente del conexionado***. A un usuario de este nivel le es indiferente que el display esté conectado a 4 bits, que el control sea en bucle abierto etc...Lo más importante es que si por alguna razón se desea cambiar el conexionado, el nivel de aplicación no

habrá que modificarlo. En los siguientes apartados se va a desarrollar el software del nivel físico para las 4 opciones de conexión. El objetivo es implementar las funciones de interfaz **dato_lcd(), control_lcd() e init_lcd().** Una vez creado este nivel, en el nivel de aplicación se desarrollarán rutinas de control del LCD de más nivel como por ejemplo impresión de cadenas, caracteres creados por el usuario...

3.2.- EL SOFTWARE DEL NIVEL FISICO

Ente apartado se presentan las implementaciones de los distintos conexionados. En todos los programas ejemplo el programa principal hace lo mismo: imprime un prompt en pantalla (C>) y todo lo recibido por el puerto serie lo imprime en el LCD.

3.2.1.- OPCION A: Bucle cerrado y bus de datos a 4 bits. Programa LCDOPA.ASM

Tamaño del programa: 141 bytes

```
; +-------------------------------------------------------------------------+
; | LCDOPA.ASM (C) GRUPO J&J. Octubre 1997.                                 |
; |-------------------------------------------------------------------------|
; |  Programa ejemplo para ser ejecutado en la tarjeta CT6811.              |
; |  Este programa se debe cargar en la RAM interna del 6811.               |
; |                                                                         |
; |    OPCION A DE CONEXION DEL LCD A LA CT6811: Bus de datos del LCD a     |
; |   4 bits y control en bucle cerrado.                                    |
; |                                                                         |
; |    CONEXIONADO CT6811-LCD                                               |
; |                                                                         |
; |      CT6811            LCD                                              |
; |                                                                         |
; |        PC0 -----------> E                                               |
; |        PC1 -----------> RS                                              |
; |        PC2 -----------> R/W                                             |
; |        PC4 -----------> DB4                                             |
; |        PC5 -----------> DB5                                             |
; |        PC6 -----------> DB6                                             |
; |        PC7 -----------> DB7                                             |
; |                                                                         |
; +-------------------------------------------------------------------------+

; +-------------------------------------------------------------------------+
; |  SOFTWARE DE NIVEL FISICO                                               |
; |                                                                         |
; |    INTERFAZ OFRECIDO A LOS NIVELES SUPERIORES:                          |
; |                                                                         |
; |    init_lcd --> Inicializar LCD a 4 bits                                |
; |    dato_lcd ---> Enviar un dato al LCD (CG RAM o DD RAM). Se envía      |
; |                  el dato que se encuentra en el acumulador A            |
; |    cmd_lcd-->   Enviar un comando al LCD. El codigo de comando se       |
; |                   encuentra en el acumulador A                          |
; |                                                                         |
; +-------------------------------------------------------------------------+

; +-------------------------------------------------------------------------+
; |  El programa principal de ejemplo simplemente escribe en el LCD         |
; |  "C>" y todo lo que se recibe por el puerto serie.                      |
; +-------------------------------------------------------------------------+

;                       +------------+
;----------------------¦ CONSTANTES ¦----------------------
;                       +------------+
PORTC   EQU $03     ; Puerto C
DDRC    EQU $07     ; Sentido de los pines del puerto C
RS      EQU $02
RW      EQU $04
E       EQU $01

;                 +--------------------+
;-----------------¦ PROGRAMA PRINCIPAL ¦----------------------
;                 +--------------------+

        ORG $0000

        LDX #$1000
        BSR init_lcd    ; Inicializar el LCD
```

```
        LDAA #'C'       ; Imprimir: "C>"
        BSR dato_lcd
        LDAA #'>'
        BSR dato_lcd

bucle
        BSR leer_car    ; Esperar a que se pulse una tecla
        BSR dato_lcd    ; Imprimir en el LCD el caracter recibido

        BRA bucle

;                +------------------------+
;----------------| FIN PROGRAMA PRINCIPAL |----------------------
;                +------------------------+


;+---------------------------------------------+
;| INIT_LCD : Inicialización del LCD a 4 bits. |
;|                                             |
;| ENTRADAS: Ninguna.                          |
;| SALIDAS: Ninguna.                           |
;+---------------------------------------------+
init_lcd
        LDAA #$FF
        STAA DDRC,X     ; Configuración PUERTO C para salida

        LDAA #$28        ; RS=0; R/W=0; E=0;
        STAA PORTC,X
        BSR enable_lcd

        LDAA #$40        ; Pequeña pausa
        BSR pausa

        LDAA #$20        ; Configurar BUS a 4 bits
        STAA PORTC,X
        BSR enable_lcd
        LDAA #$80
        STAA PORTC,X
        BSR enable_lcd

        LDAA #$0E        ; Display ON
        BSR cmd_lcd

        LDAA #$01        ; CLS
        BSR cmd_lcd
        RTS

;+-----------------------------------------+
;| CMD_LCD: Enviar un comando al LCD.      |
;|                                         |
;| ENTRADAS: A Contiene el comando a enviar|
;| SALIDAS: Ninguna.                       |
;+-----------------------------------------+
cmd_lcd
        BSR busy         ; Esperar hasta que LCD esté libre
        BCLR PORTC,X $06 ; RS=0; R/W=0;
        BSR enviar_lcd   ; Mandar comando
        RTS

;+-------------------------------------------------------+
;| DATO_LCD : Enviar un dato al LCD a la CG RAM o DD RAM |
;|                                                       |
;| ENTRADAS: A contiene el dato a enviar al LCD          |
;| SALIDAS: Ninguna.                                     |
;+-------------------------------------------------------+
dato_lcd
        BSR busy       ; Esperar a que LCD esté libre
        LDAB #$02      ; RS=1,  R/W=0
        STAB PORTC,X
        BSR enviar_lcd ; Mandar dato
        RTS

; --------------- FUNCIONES EMPLEADAS PARA LA IMPLEMENTACION ----------

;+-----------------------------------------+
;| PAUSA: Realizar una pausa.              |
;|                                         |
;| ENTRADAS: A contiene la pausa a realizar|
;| SALIDAS: Ninguna.                       |
;+-----------------------------------------+
```

```
pausa   DECA
        BNE   pausa
        RTS

;+---------------------------------------------------------+
;¦ ENVIAR_LCD: Enviar un byte al LCD. Las señales RS y     ¦
;¦             R/W no se modifican.                        ¦
;¦ ENTRADAS: A contiene el byte a enviar                   ¦
;¦ SALIDAS: Ninguna.                                       ¦
;+---------------------------------------------------------+
enviar_lcd
        BSR enviar_4bits    ; Enviar los 4 bits más significativos
        ROLA                ; Desplazar A 4 bits a la izquierda
        ROLA
        ROLA
        ROLA
        BSR enviar_4bits    ; Enviar los 4 bits menos significativos
        RTS

;+---------------------------------------------------------+
;¦ ENVIAR_4BITS: Depositar 4 bits en el bus de datos y     ¦
;¦               mandar un pulso en la señal E             ¦
;¦ ENTRADAS: Los 4 bits mas significativos del acumulador  ¦
;¦           A son los que se envían al LCD.               ¦
;¦ SALIDAS: Ninguna.                                       ¦
;+---------------------------------------------------------+
enviar_4bits
        PSHA
        LDAB PORTC,X    ; Leer en B el puerto C
        ANDB #$0F       ; Borrar de B los 4 bits mas significativos
        ANDA #$F0       ; Borrar de A los 4 bits menos significatos
        ABA             ; A=A+B
        STAA PORTC,X    ; Enviar dato al puerto C
        BSR enable_lcd  ; Mandar un pulso en la señal E
        PULA
        RTS

;+---------------------------------------------------------+
;¦ ENABLE_LCD: Enviar un pulso por la señal E del LCD para ¦
;¦             validar los datos.                          ¦
;¦ ENTRADAS: Ninguna.                                      ¦
;¦ SALIDAS: Ninguna.                                       ¦
;+---------------------------------------------------------+
enable_lcd
        BSET PORTC,X E  ; E=1
        BCLR PORTC,X E  ; E=0
        RTS

;+---------------------------------------------------------+
;¦ BUSY_LCD: Esperar hasta que el LCD esté libre.          ¦
;¦ ENTRADAS: Ninguna.                                      ¦
;¦ SALIDAS: Ninguna.                                       ¦
;+---------------------------------------------------------+
busy     LDAB #$0F         ; Configurar bits PC7,PC6,PC5,PC4 de entrada
         STAB DDRC,X       ;
         BSET PORTC,X RW   ; R/W=1
         BCLR PORTC,X RS   ; RS=0
wait_lcd
         BSET PORTC,X E  ; E=1
         LDAB PORTC,X    ; Leer byte mas significativo
         BCLR PORTC,X E  ; E=0
         BSR enable_lcd  ; El byte menos significativo no interesa
         ANDB #$80       ; Comprobar el bit 7 del byte leido.
         BNE wait_lcd    ; Si es igual a 1 el LCD está ocupado

         LDAB #$FF
         STAB DDRC,X       ; Configurar PUERTO C para salida
         RTS

;------------------- RUTINA PARA LAS COMUNICACIONES SERIE --------------

SCSR    equ  $2E
SCDR    equ  $2F

leer_car BRCLR SCSR,X $20 leer_car   ; Esperar hasta que llegue un carácter
         LDAA SCDR,X
         RTS
```

3.2.2.- OPCION B: Bucle cerrado y bus de datos a 8 bits. Programa LCDOPB.ASM

Tamaño del programa: 99 Bytes

```
; +--------------------------------------------------------------------------+
; | LCDOPB.ASM (C) GRUPO J&J. Octubre 1997.                                  |
; |--------------------------------------------------------------------------|
; |  Programa ejemplo para ser ejecutado en la tarjeta CT6811.               |
; |  Este programa se debe cargar en la RAM interna del 6811.                |
; |                                                                          |
; |    OPCION B DE CONEXION DEL LCD A LA CT6811: Bus de datos del LCD a      |
; |   8 bits y control en bucle cerrado.                                     |
; |                                                                          |
; |    CONEXIONADO CT6811-LCD                                                |
; |                                                                          |
; |      CT6811            LCD                                               |
; |                                                                          |
; |        PB0 -----------> E                                                |
; |        PB1 -----------> RS                                               |
; |        PB2 -----------> R/W                                              |
; |                                                                          |
; |        PC0 -----------> DB0                                              |
; |        PC1 -----------> DB1                                              |
; |        PC2 -----------> DB2                                              |
; |        PC3 -----------> DB3                                              |
; |        PC4 -----------> DB4                                              |
; |        PC5 -----------> DB5                                              |
; |        PC6 -----------> DB6                                              |
; |        PC7 -----------> DB7                                              |
; |                                                                          |
; +--------------------------------------------------------------------------+

; +--------------------------------------------------------------------------+
; |  SOFTWARE DE NIVEL FISICO                                                |
; |                                                                          |
; |    INTERFAZ OFRECIDO A LOS NIVELES SUPERIORES:                           |
; |                                                                          |
; |    init_lcd --> Inicializar LCD a 4 bits                                 |
; |    dato_lcd ---> Enviar un dato al LCD (CG RAM o DD RAM). Se envía       |
; |                  el dato que se encuentra en el acumulador A             |
; |    cmd_lcd-->   Enviar un comando al LCD. El codigo de comando se        |
; |                   encuentra en el acumulador A                           |
; |                                                                          |
; +--------------------------------------------------------------------------+

; +--------------------------------------------------------------------------+
; |  El programa principal de ejemplo simplemente escribe en el LCD          |
; |  "C>" y todo lo que se recibe por el puerto serie.                       |
; +--------------------------------------------------------------------------+

;                       +------------+
;----------------------¦ CONSTANTES ¦----------------------
;                       +------------+
PORTC   EQU $03     ; Puerto C
PORTB   EQU $04     ; Puerto B
DDRC    EQU $07     ; Sentido de los pines del puerto C
RS      EQU $02
RW      EQU $04
E       EQU $01

;                 +--------------------+
;-----------------¦ PROGRAMA PRINCIPAL ¦----------------------
;                 +--------------------+

        ORG $0000

        LDX #$1000
        BSR init_lcd    ; Inicializar el LCD

        LDAA #'C'       ; Imprimir: "C>"
        BSR dato_lcd
        LDAA #'>'
        BSR dato_lcd

bucle
        BSR leer_car    ; Esperar a que se pulse una tecla
        BSR dato_lcd    ; Imprimir en el LCD el caracter recibido

        BRA bucle
```

```
;                +------------------------+
;----------------| FIN PROGRAMA PRINCIPAL |----------------------
;                +------------------------+

;+---------------------------------------------+
;| INIT_LCD : Inicialización del LCD a 4 bits. |
;|                                             |
;| ENTRADAS: Ninguna.                          |
;| SALIDAS: Ninguna.                           |
;+---------------------------------------------+
init_lcd
        LDAA #$FF
        STAA DDRC,X     ; Configuración PUERTO C para salida

        LDAA #$38       ;
        BSR cmd_lcd     ; Inicializar display a 8 bits

        LDAA #$0E       ; Display ON
        BSR cmd_lcd

        LDAA #$01       ; CLD
        BSR cmd_lcd
        RTS

;+-----------------------------------------+
;| CMD_LCD: Enviar un comando al LCD.      |
;|                                         |
;| ENTRADAS: A Contiene el comando a enviar|
;| SALIDAS: Ninguna.                       |
;+-----------------------------------------+
cmd_lcd BSR busy
        STAA PORTC,X
        BCLR PORTB,X $06 ; RS=0; R/W=0;
        BSR enable_lcd
        RTS

;+---------------------------------------------------------+
;| DATO_LCD : Enviar un dato al LCD a la CG RAM o DD RAM   |
;|                                                         |
;| ENTRADAS: A contiene el dato a enviar al LCD            |
;| SALIDAS: Ninguna.                                       |
;+---------------------------------------------------------+
dato_lcd
        BSR busy
        BSET PORTB,X RS ; RS=1;
        BCLR PORTB,X RW ; R/W=0;
        STAA PORTC,X
        BSR enable_lcd
        RTS

; --------------- FUNCIONES EMPLEADAS PARA LA IMPLEMENTACION ----------

;+---------------------------------------------------------+
;| ENABLE_LCD: Enviar un pulso por la señal E del LCD para |
;|             validar los datos.                          |
;| ENTRADAS: Ninguna.                                      |
;| SALIDAS: Ninguna.                                       |
;+---------------------------------------------------------+
enable_lcd
        BSET PORTB,X E  ; E=1
        BCLR PORTB,X E  ; E=0
        RTS

;+---------------------------------------------------------+
;| BUSY_LCD: Esperar hasta que el LCD esté libre.          |
;| ENTRADAS: Ninguna.                                      |
;| SALIDAS: Ninguna.                                       |
;+---------------------------------------------------------+
busy     PSHB
         CLRB
         STAB DDRC,X       ;Configuración PUERTO C para entrada
         BSET PORTB,X RW   ; R/W=1
         BCLR PORTB,X RS   ; RS=0
         BSET PORTB,X E
wait_lcd BRSET PORTC,X $80 wait_lcd
         BCLR PORTB,X E
         LDAB #$FF
         STAB DDRC,X       ; Configurar PUERTO C para salida
         PULB
         RTS
```

```
;------------------- RUTINA PARA LAS COMUNICACIONES SERIE --------------

SCSR    equ  $2E
SCDR    equ  $2F

leer_car BRCLR SCSR,X $20 leer_car   ; Esperar hasta que llegue un carácter
         LDAA SCDR,X
         RTS
```

3.2.3.- OPCION C: Bucle abierto y bus de datos a 4 bits

Tamaño: 143 Bytes

```
; +-------------------------------------------------------------------------+
; | LCDOPC.ASM (C) GRUPO J&J. Octubre 1997.                                 |
; |-------------------------------------------------------------------------|
; |  Programa ejemplo para ser ejecutado en la tarjeta CT6811.              |
; |  Este programa se debe cargar en la RAM interna del 6811.               |
; |                                                                         |
; |    OPCION C DE CONEXION DEL LCD A LA CT6811: Bus de datos del LCD a     |
; |   4 bits y control en bucle abierto.                                    |
; |                                                                         |
; |    CONEXIONADO CT6811-LCD                                               |
; |                                                                         |
; |      CT6811            LCD                                              |
; |                                                                         |
; |        PC0 -----------> E                                               |
; |        PC1 -----------> RS                                              |
; |                         R/W (a masa siempre)                            |
; |        PC4 -----------> DB4                                             |
; |        PC5 -----------> DB5                                             |
; |        PC6 -----------> DB6                                             |
; |        PC7 -----------> DB7                                             |
; |                                                                         |
; +-------------------------------------------------------------------------+

; +-------------------------------------------------------------------------+
; |  SOFTWARE DE NIVEL FISICO                                               |
; |                                                                         |
; |    INTERFAZ OFRECIDO A LOS NIVELES SUPERIORES:                          |
; |                                                                         |
; |    init_lcd --> Inicializar LCD a 4 bits                                |
; |    dato_lcd ---> Enviar un dato al LCD (CG RAM o DD RAM). Se envía      |
; |                  el dato que se encuentra en el acumulador A            |
; |    cmd_lcd-->   Enviar un comando al LCD. El codigo de comando se       |
; |                   encuentra en el acumulador A                          |
; |                                                                         |
; +-------------------------------------------------------------------------+

; +-------------------------------------------------------------------------+
; |  El programa principal de ejemplo simplemente escribe en el LCD         |
; |  "C>" y todo lo que se recibe por el puerto serie.                      |
; +-------------------------------------------------------------------------+

;                      +------------+
;---------------------¦ CONSTANTES ¦----------------------
;                      +------------+
PORTC    EQU $03     ; Puerto C
DDRC     EQU $07     ; Sentido de los pines del puerto C
RS       EQU $02
RW       EQU $04
E        EQU $01
PAUSACLS EQU $1000
PAUSALCD EQU $20

;                 +--------------------+
;-----------------¦ PROGRAMA PRINCIPAL ¦----------------------
;                 +--------------------+

        ORG $0000

        LDX #$1000
        BSR init_lcd    ; Inicializar el LCD

        LDAA #'C'       ; Imprimir: "C>"
        BSR dato_lcd
        LDAA #'>'
        BSR dato_lcd
```

```
bucle
        BSR leer_car    ; Esperar a que se pulse una tecla
        BSR dato_lcd    ; Imprimir en el LCD el caracter recibido

        BRA bucle

;                +------------------------+
;----------------| FIN PROGRAMA PRINCIPAL |----------------------
;                +------------------------+

;+--------------------------------------------------------+
;| DATO_LCD : Enviar un dato al LCD a la CG RAM o DD RAM  |
;|                                                        |
;| ENTRADAS: A contiene el dato a enviar al LCD           |
;| SALIDAS: Ninguna.                                      |
;+--------------------------------------------------------+
dato_lcd
        LDAB #$02       ; RS=1,  R/W=0
        STAB PORTC,X
        BSR enviar_lcd ; Mandar dato
        LDY #$07
        BSR pausa       ; Realizar una pausa de 40microseg.
        RTS

SCSR    equ  $2E
SCDR    equ  $2F

;+---------------------------------------------+
;| INIT_LCD : Inicialización del LCD a 4 bits. |
;|                                             |
;| ENTRADAS: Ninguna.                          |
;| SALIDAS: Ninguna.                           |
;+---------------------------------------------+
init_lcd
        LDAA #$FF
        STAA DDRC,X     ; Configuración PUERTO C para salida

        LDAA #$28       ; RS=0; R/W=0; E=0;
        STAA PORTC,X
        BSR enable_lcd

        LDY #PAUSALCD   ; Pequeña pausa
        BSR pausa

        LDAA #$20       ; Configurar BUS a 4 bits
        STAA PORTC,X
        BSR enable_lcd
        LDAA #$80
        STAA PORTC,X
        BSR enable_lcd

        LDY #PAUSALCD   ; Pequeña pausa
        BSR pausa

        LDAA #$0F        ; Display ON
        BSR cmd_lcd

        LDAA #$01        ; CLS
        BSR cmd_lcd
        RTS

;+-----------------------------------------+
;| CMD_LCD: Enviar un comando al LCD.      |
;|                                         |
;| ENTRADAS: A Contiene el comando a enviar|
;| SALIDAS: Ninguna.                       |
;+-----------------------------------------+
cmd_lcd
        PSHA
        BCLR PORTC,X $06 ; RS=0; R/W=0;
        BSR enviar_lcd   ; Mandar comando
        PULA
        ;-- Para los comandos home y cls hay que hacer pausa de 1.67ms
        ;-- Para el resto 40microseg.
        LDY #PAUSALCD
        ANDA #$FC
        BNE sigue       ; Si comando que no es ni CLS ni HOME
        LDY #PAUSACLS
sigue
        BSR pausa
```

```
        RTS

; -------------- FUNCIONES EMPLEADAS PARA LA IMPLEMENTACION ----------

;+-----------------------------------------------+
;¦ PAUSA: Realizar una pausa.                    ¦
;¦                                               ¦
;¦ ENTRADAS: Y contiene la pausa a realizar      ¦
;¦           Cada unidad de pausa son 6microseg  ¦
;¦ SALIDAS: Ninguna.                             ¦
;+-----------------------------------------------+
pausa   DEY
        CPY #0
        BNE pausa
        RTS

;+-------------------------------------------------------+
;¦ ENVIAR_LCD: Enviar un byte al LCD. Las señales RS y    ¦
;¦             R/W no se modifican.                      ¦
;¦ ENTRADAS: A contiene el byte a enviar                 ¦
;¦ SALIDAS: Ninguna.                                     ¦
;+-------------------------------------------------------+
enviar_lcd
        BSR enviar_4bits    ; Enviar los 4 bits más significativos
        ROLA                ; Desplazar A 4 bits a la izquierda
        ROLA
        ROLA
        ROLA
        BSR enviar_4bits    ; Enviar los 4 bits menos significativos
        RTS

;+-------------------------------------------------------+
;¦ ENVIAR_4BITS: Depositar 4 bits en el bus de datos y    ¦
;¦               mandar un pulso en la señal E           ¦
;¦ ENTRADAS: Los 4 bits mas significativos del acumulador ¦
;¦           A son los que se envían al LCD.             ¦
;¦ SALIDAS: Ninguna.                                     ¦
;+-------------------------------------------------------+
enviar_4bits
        PSHA
        LDAB PORTC,X    ; Leer en B el puerto C
        ANDB #$0F       ; Borrar de B los 4 bits mas significativos
        ANDA #$F0       ; Borrar de A los 4 bits menos significatos
        ABA             ; A=A+B
        STAA PORTC,X    ; Enviar dato al puerto C
        BSR enable_lcd  ; Mandar un pulso en la señal E
        PULA
        RTS

;+-------------------------------------------------------+
;¦ ENABLE_LCD: Enviar un pulso por la señal E del LCD para¦
;¦             validar los datos.                        ¦
;¦ ENTRADAS: Ninguna.                                    ¦
;¦ SALIDAS: Ninguna.                                     ¦
;+-------------------------------------------------------+
enable_lcd
        BSET PORTC,X E  ; E=1
        BCLR PORTC,X E  ; E=0
        RTS

;------------------- RUTINA PARA LAS COMUNICACIONES SERIE --------------
leer_car BRCLR SCSR,X $20 leer_car   ; Esperar hasta que llegue un carácter
         LDAA SCDR,X
         RTS
```

3.2.4.- OPCION D: Bucle abierto y bus de datos a 8 bits

```
; +-------------------------------------------------------------------------+
; | LCDOPD.C (C) GRUPO J&J. Octubre 1997.                                   |
; |-------------------------------------------------------------------------|
; |  Programa ejemplo para ser ejecutado en la tarjeta CT6811.              |
; |  Este programa se debe cargar en la RAM interna del 6811.               |
; |                                                                         |
; |    OPCION D DE CONEXION DEL LCD A LA CT6811: Bus de datos del LCD a     |
; |   8 bits y control en bucle abierto.                                    |
; |                                                                         |
; |    CONEXIONADO CT6811-LCD                                               |
; |                                                                         |
; |      CT6811            LCD                                              |
; |                                                                         |
; |        PB0 -----------> E                                               |
; |        PB1 -----------> RS                                              |
; |                         R/W (A masa)                                    |
; |                                                                         |
; |        PC0 -----------> DB0                                             |
; |        PC1 -----------> DB1                                             |
; |        PC2 -----------> DB2                                             |
; |        PC3 -----------> DB3                                             |
; |        PC4 -----------> DB4                                             |
; |        PC5 -----------> DB5                                             |
; |        PC6 -----------> DB6                                             |
; |        PC7 -----------> DB7                                             |
; |                                                                         |
; +-------------------------------------------------------------------------+

; +-------------------------------------------------------------------------+
; |  SOFTWARE DE NIVEL FISICO                                               |
; |                                                                         |
; |    INTERFAZ OFRECIDO A LOS NIVELES SUPERIORES:                          |
; |                                                                         |
; |    init_lcd --> Inicializar LCD a 4 bits                                |
; |    dato_lcd ---> Enviar un dato al LCD (CG RAM o DD RAM). Se envía      |
; |                  el dato que se encuentra en el acumulador A            |
; |    cmd_lcd-->   Enviar un comando al LCD. El codigo de comando se       |
; |                   encuentra en el acumulador A                          |
; |                                                                         |
; +-------------------------------------------------------------------------+

; +-------------------------------------------------------------------------+
; |  El programa principal de ejemplo simplemente escribe en el LCD         |
; |  "C>" y todo lo que se recibe por el puerto serie.                      |
; +-------------------------------------------------------------------------+

;                      +------------+
;---------------------¦ CONSTANTES ¦----------------------
;                      +------------+
PORTC   EQU $03     ; Puerto C
PORTB   EQU $04     ; Puerto B
DDRC    EQU $07     ; Sentido de los pines del puerto C
RS      EQU $02
RW      EQU $04
E       EQU $01
PAUSACLS EQU $1000
PAUSALCD EQU $20


;                 +--------------------+
;-----------------¦ PROGRAMA PRINCIPAL ¦----------------------
;                 +--------------------+

        ORG $0000

        LDX #$1000
        BSR init_lcd    ; Inicializar el LCD

        LDAA #'C'       ; Imprimir: "C>"
        BSR dato_lcd
        LDAA #'>'
        BSR dato_lcd

bucle
        BSR leer_car    ; Esperar a que se pulse una tecla
        BSR dato_lcd    ; Imprimir en el LCD el caracter recibido

        BRA bucle
```

```
;                +------------------------+
;----------------| FIN PROGRAMA PRINCIPAL |----------------------
;                +------------------------+

;+---------------------------------------------+
;| INIT_LCD : Inicialización del LCD a 4 bits. |
;|                                             |
;| ENTRADAS: Ninguna.                          |
;| SALIDAS: Ninguna.                           |
;+---------------------------------------------+
init_lcd
        LDAA #$FF
        STAA DDRC,X     ; Configuración PUERTO C para salida

        LDAA #$38       ;
        BSR cmd_lcd     ; Inicializar display a 8 bits

        LDAA #$0E       ; Display ON
        BSR cmd_lcd

        LDAA #$01       ; CLD
        BSR cmd_lcd
        RTS

;+-----------------------------------------+
;| CMD_LCD: Enviar un comando al LCD.      |
;|                                         |
;| ENTRADAS: A Contiene el comando a enviar|
;| SALIDAS: Ninguna.                       |
;+-----------------------------------------+
cmd_lcd STAA PORTC,X
        BCLR PORTB,X $06 ; RS=0; R/W=0;
        BSR enable_lcd
        ;-- Para los comandos home y cls hay que hacer pausa de 1.67ms
        ;-- Para el resto 40microseg.
        LDY #PAUSALCD
        ANDA #$FC
        BNE sigue       ; Si comando que no es ni CLS ni HOME
        LDY #PAUSACLS
sigue
        BSR pausa
        RTS

;+---------------------------------------------------------+
;| DATO_LCD : Enviar un dato al LCD a la CG RAM o DD RAM   |
;|                                                         |
;| ENTRADAS: A contiene el dato a enviar al LCD            |
;| SALIDAS: Ninguna.                                       |
;+---------------------------------------------------------+
dato_lcd
        BSET PORTB,X RS ; RS=1;
        BCLR PORTB,X RW ; R/W=0;
        STAA PORTC,X
        BSR enable_lcd
        LDY #PAUSALCD
        BRA pausa
        RTS

; --------------- FUNCIONES EMPLEADAS PARA LA IMPLEMENTACION ----------

;+---------------------------------------------------------+
;| ENABLE_LCD: Enviar un pulso por la señal E del LCD para |
;|             validar los datos.                          |
;| ENTRADAS: Ninguna.                                      |
;| SALIDAS: Ninguna.                                       |
;+---------------------------------------------------------+
enable_lcd
        BSET PORTB,X E  ; E=1
        BCLR PORTB,X E  ; E=0
        RTS

;+-----------------------------------------+
;| PAUSA: Realizar una pausa.              |
;|                                         |
;| ENTRADAS: Y contiene la pausa a realizar|
;|           Cada unidad de pausa son 6microseg|
;| SALIDAS: Ninguna.                       |
;+-----------------------------------------+
pausa   DEY
        CPY #0
```

```
        BNE pausa
        RTS


;------------------- RUTINA PARA LAS COMUNICACIONES SERIE --------------

SCSR    equ  $2E
SCDR    equ  $2F

leer_car BRCLR SCSR,X $20 leer_car   ; Esperar hasta que llegue un carácter
         LDAA SCDR,X
         RTS
```

3.3.- SOFTWARE DEL NIVEL DE APLICACION

En esta sección se van a presentar programas que haciendo uso de las rutinas ofrecidas por el nivel físico: init_lcd(), dato_lcd() y cmd_lcd() implementan funciones de mayor nivel como impresión de cadenas y definición de caracteres por el usuario. No se presentan los programas completos. A cada ejemplo habrá que añadir el software del nivel físico según el tipo de conexionado empelado.

3.1.- Impresión de cadenas en el LCD.

```
; +-------------------------------------------------------------------------+
; ¦ PRINTLCD.ASM (C) GRUPO J&J. Octubre 1997.                               ¦
; ¦-------------------------------------------------------------------------¦
; ¦  Programa ejemplo para ser ejecutado en la tarjeta CT6811.              ¦
; ¦  Este programa se debe cargar en la RAM interna del 6811.               ¦
; ¦                                                                         ¦
; ¦    Ejemplo de manejo de las rutinas de nivel físico del LCD. A partir   ¦
; ¦   de las subrutinas del nivel fisico: init_lcd, cmd_lcd y dato_lcd       ¦
; ¦   se crea un servicio de nivel superio de impresion de cadenas en el    ¦
; ¦   LCD.                                                                  ¦
; ¦                                                                         ¦
; ¦     El tipo de conexionado elegido es la OPCION C, pero las rutinas     ¦
; ¦   desarrolladas en el ejemplo sirven para cualquier opción, puesto      ¦
; ¦   que son independientes de cómo esté implementado el nivel físico      ¦
; ¦                                                                         ¦
; +-------------------------------------------------------------------------+

; +-------------------------------------------------------------------------+
; ¦  El programa principal simplemente escribe dos cadenas en las dos       ¦
; ¦  líneas del LCD.                                                        ¦
; +-------------------------------------------------------------------------+

;                +--------------------+
;----------------¦ PROGRAMA PRINCIPAL ¦----------------------
;                +--------------------+
        ORG $0000

        LDX #$1000
        BSR init_lcd    ; Inicializar el LCD

        LDY #cad1       ; Imprimir cadena
        BSR print_lcd

        LDAA #$C0       ; Situar cursor en posición (1,2)
        BSR cmd_lcd

        LDY #cad2
        BSR print_lcd  ; Imprimir cadena
inf     BRA inf

;                +------------------------+
;----------------¦ FIN PROGRAMA PRINCIPAL ¦----------------------
;                +------------------------+

;+---------------------------------------------------------+
;¦ Enviar una cadena de caracteres al LCD.                 ¦
;¦ La cadena debe terminar con el carácter 0               ¦
;¦ ENTRADAS: Registro Y contiene dirección cadena a enviar ¦
;¦ SALIDAS: Ninguna                                        ¦
;+---------------------------------------------------------+
print_lcd
         LDAA 0,Y        ; Meter en A el carácter a enviar
         CMPA #0         ; ¿Fin de la cadena?
```

```
        BEQ finp        ; Si--> retornar
        PSHY
        BSR dato_lcd    ; NO--> enviar carácter.
        PULY
        INY             ; Apuntar a la sig. posición de memoria
        BRA print_lcd   ; Repetir todo
finp    RTS

;+--------+
;¦ DATOS  ¦
;+--------+
cad1    FCC " PRUEBA DEL LCD"
        FCB 0
cad2    FCC "*(C) GRUPO J&J*"
        FCB 0

;                +--------------------------+
;----------------¦ SOFTWARE DE NIVEL FISICO ¦----------------------
;                +--------------------------+
;       Aqui hay que incluir la implementacion de las rutinas:
;   init_lcd(), dato_lcd() y cmd_lcd(), según la opcion de conexión empleada
```

3.2.- Definición de caracteres

En este ejemplo se definen dos caracteres y se imprimen. Obsérverse que para definir los caracteres se ha utilizado la misma rutina que para imprimir cadenas.

```
; +-------------------------------------------------------------------------+
; ¦ DEFCAR.ASM   (C) GRUPO J&J. Octubre 1997.                               ¦
; ¦-------------------------------------------------------------------------¦
; ¦  Programa ejemplo para ser ejecutado en la tarjeta CT6811.              ¦
; ¦  Este programa se debe cargar en la RAM interna del 6811.               ¦
; ¦                                                                         ¦
; ¦    Ejemplo de manejo de las rutinas de nivel físico del LCD. A partir   ¦
; ¦   de las subrutinas del nivel fisico: init_lcd, cmd_lcd y dato_lcd       ¦
; ¦   se crea un servicio de nivel superio de definición de caracteres      ¦
; ¦   por el usuario.                                                       ¦
; ¦                                                                         ¦
; ¦     El tipo de conexionado elegido es la OPCION C, pero las rutinas     ¦
; ¦   desarrolladas en el ejemplo sirven para cualquier opción, puesto      ¦
; ¦   que son independientes de cómo esté implementado el nivel físico     ¦
; ¦                                                                         ¦
; +-------------------------------------------------------------------------+

; +-------------------------------------------------------------------------+
; ¦ El programa principal dibuja una llave y un monigote en el LCD          ¦
; +-------------------------------------------------------------------------+

;                +--------------------+
;----------------¦ PROGRAMA PRINCIPAL ¦----------------------
;                +--------------------+
        ORG $0000

        LDX #$1000
        BSR init_lcd    ; Inicializar el LCD

        LDAA #$40       ; Seleccionar direccion 0 de la CG RAM
        BSR cmd_lcd

        LDY #car0       ; Definir el caracter 0
        BSR print_lcd
        LDY #car1       ; Definir el carácter 1
        BSR print_lcd

        LDAA #$80       ; Seleccionar dirección 0 de la DD RAM
        BSR cmd_lcd

        LDAA #$00       ; Imprimir una llave
        BSR dato_lcd

        LDAA #$01
        BSR dato_lcd    ; Imprimir un monigote

inf     BRA inf

;                +------------------------+
;----------------¦ FIN PROGRAMA PRINCIPAL ¦----------------------
```

```
;                +------------------------+

;+---------------------------------------------------------+
;¦ Enviar una cadena de caracteres al LCD.                 ¦
;¦ La cadena debe terminar con el carácter 0               ¦
;¦ ENTRADAS: Registro Y contiene dirección cadena a enviar ¦
;¦ SALIDAS: Ninguna                                        ¦
;+---------------------------------------------------------+
print_lcd
         LDAA 0,Y        ; Meter en A el carácter a enviar
         CMPA #0         ; ¿Fin de la cadena?
         BEQ finp        ; Si--> retornar
         PSHY
         BSR dato_lcd    ; NO--> enviar carácter.
         PULY
         INY             ; Apuntar a la sig. posición de memoria
         BRA print_lcd   ; Repetir todo
finp     RTS

;+--------+
;¦ DATOS  ¦
;+--------+
car0     DB %01110       ; Una llave
         DB %10001
         DB %01110
         DB %00100
         DB %00100
         DB %11100
         DB %00100
         DB %11100
         FCB 0

car1     DB %01110       ; Un monigote
         DB %01110
         DB %00100
         DB %11111
         DB %00100
         DB %00100
         DB %01010
         DB %10001
         FCB 0

;                +--------------------------+
;----------------¦ SOFTWARE DE NIVEL FISICO ¦----------------------
;                +--------------------------+
;       Aqui hay que incluir la implementacion de las rutinas:
;   init_lcd(), dato_lcd() y cmd_lcd(), según la opcion de conexión empleada
```

4.- PROGRAMACION DEL LCD A TRAVES DEL CTSERVER

4.1.- INTRODUCCION

El software se sigue dividiendo en dos niveles: **nivel físico**, que depende del tipo de conexionado y **nivel de aplicacion** que se apoya en las rutinas de interfaz del nivel fisico y construye rutinas de mayor nivel. Al programar bajo el CTSERVER ya no tiene sentido hacerlo en bucle cerrado puesto que la velocidad está determinada por la velocidad de comunicacione entre cliente y servidor. Por ello sólo son interesantes la opciones C y D.

4.2.- SOFTWARE DE NIVEL FISICO

4.2.1.- OPCION C: Bucle abierto y LCD a 4 bits

Sólo se presentan las rutinas por separado y no un programa completo. El programa completo se encuentra en la sección 4.3.

```
/*          +-----------------------------------------+
            ¦     RUTINAS DE NIVEL FISICO DEL LCD.    ¦
            ¦ La implementación de estas rutinas depende¦
            ¦ de la opción de conexionado.            ¦
            +-----------------------------------------+              */


/*--- INTERFAZ ---*/

void dato_lcd(byte dat);
void cmd_lcd(byte cod);
void init_lcd();

/*--- Rutinas de implementación --*/

void enviar_lcd(byte dat);


void init_lcd()
/*
+-----------------------------+
¦ Inicializar el LCD a 8 bits. ¦
+-----------------------------+*/
{
  store(0xFF,DDRC);   // Configurar PUERTO C para salida
  store(0x00,PORTC);

  enviar_lcd(0x30);   // Configurar para 8 bits
  enviar_lcd(0x80);   // Relleno...

  enviar_lcd(0x20);

  enviar_lcd(0x20);
  enviar_lcd(0x80);

  cmd_lcd(control_lcd);  // Encender display y cursor
  cmd_lcd(0x01);         // CLS

}

void cmd_lcd(byte cod)
/*
+-----------------------------+
¦ Ejecutar un comando del LCD ¦
+-----------------------------+*/
{
   puertoc=0x00;               // RS=0; R/W=0;
   enviar_lcd(cod&0xF0);       // Enviar 4 bits mas signfic. con RS=0, R/W=0
   enviar_lcd((cod<<4)&0xF0);  // Enviar 4 bits menos signif.
}

void dato_lcd(byte dat)
/*
+-----------------------------+
¦  Enviar un carácter al LCD  ¦
+-----------------------------+*/
{
```

```
  puertoc=0x02;             // RS=1
  enviar_lcd(dat&0xF0);
  enviar_lcd((dat<<4)&0xF0);
}

void enviar_lcd(byte dat)
{
  puertoc&=0x0F;
  puertoc|=(dat & 0xF0);
  store(puertoc,PORTC);
  puertoc|=0x01;
  store(puertoc,PORTC);   // E=1
  puertoc&=~0x01;
  store(0x00,PORTC);      // E=0
}
```

4.2.2 OPCION D: Bucle abierto y bus de datos a 8 bits

Igual que en el caso anterior sólo se presentan las rutinas aisladas. El programa completo está en la sección 4.3.

```
/*          +-------------------------------------------+
            ¦     RUTINAS DE NIVEL FISICO DEL LCD.      ¦
            ¦ La implementación de estas rutinas depende¦
            ¦ de la opción de conexionado.              ¦
            +-------------------------------------------+              */


/*--- INTERFAZ ---*/

void dato_lcd(byte dat);
void cmd_lcd(byte cod);
void init_lcd();

void init_lcd()
/*
+-----------------------------+
¦ Inicializar el LCD a 8 bits. ¦
+-----------------------------+*/
{
  store(0xFF,DDRC);      // Configurar PUERTO C para salida
  store(0x00,PORTB);     // RS=0; R/W=0;
  cmd_lcd(0x38);         // Inicializar display a 8 bits
  cmd_lcd(control_lcd);  // Encender display y cursor
  cmd_lcd(0x01);         // CLS
}

void cmd_lcd(byte cod)
/*
+-----------------------------+
¦ Ejecutar un comando del LCD ¦
+-----------------------------+*/
{
   store(cod,PORTC);   // Enviar dato
   store(0x00,PORTB);
   store(0x01,PORTB);  // E=1
   store(0x00,PORTB);  // E=0
}

void dato_lcd(byte dat)
/*
+-----------------------------+
¦  Enviar un carácter al LCD  ¦
+-----------------------------+*/
{
  store(dat,PORTC);
  store(0x02,PORTB);   // RS=1, R/W=0
  store(0x03,PORTB);   // E=1
  store(0x02,PORTB);   // E=0
}
```

4.3.- SOFTWARE DE APLICACION

El programa siguiente muestra un menu de opciones para que el usuario envié al LCD los comandos que quiera. Se incluye también una pequeña animación con caracteres definidos y una rutina de impresión de cadenas. Los programas son **CTLCD4.C** para la **opción C** y **CTLCD8.C** para la **opción d.** Sólo se imprime el listado del programa CTLCD4.C. Lo único en que se diferencian es en las rutinas del nivel físico.

```
       /*
 +-------------------------------------------------------------------------+
 | CTLCD4.C    (c) GRUPO J&J. Octubre 1997.                                |
 |-------------------------------------------------------------------------|
 |                                                                         |
 | Programa cliente para el CTSERVER. CONTROL DEL LCD conectado mediante   |
 | la OPCION C: Bus de datos del LCD a 4 bits y control en bucle abierto   |
 |                                                                         |
 |     CT6811        LCD                                                   |
 |     ------        ---                                                   |
 |                                                                         |
 |     PC0 ----------> E                                                   |
 |     PC1 ----------> RS                                                  |
 |     PC2 ----------> R/W                                                 |
 |                                                                         |
 |     PC4 ----------> DB4                                                 |
 |     PC5 ----------> DB5                                                 |
 |     PC6 ----------> DB6                                                 |
 |     PC7 ----------> DB7                                                 |
 |                                                                         |
 |                                                                         |
 |     El programa principal muestra un menu de opciones en el que se      |
 | demuestra el funcionamiento de todos los comandos del LCD.              |
 |                                                                         |
 +-------------------------------------------------------------------------+
*/

#include "stdio.h"
#include "dos.h"
#include "conio.h"
#include "ctype.h"

/* ---- Librerias del Grupo J&J --- */

#include "serie.h"
#include "ctclient.h"
#include "bootstrp.h"
#include "R6811PC.H"

typedef unsigned short int byte;

byte puertoc;

/* Variables para el LCD */

byte control_lcd;    // -- Comando de control ON/OFF --
                     // -- 0 0 0 0 1 D C B
                     // D=1 Display ON. D=0 Display OFF
                     // C=1 Cursor ON. C=0 Cursor OFF
                     // B=1 Parpadeo.  B=0 No parpadeo
byte shift_lcd;      // --- Comando de desplazamiento del display ---
                     // --- 0 0 0 1 S/C R/L
                     // S/C=1 Desplazar display S/C=0 Desplazar cursor
                     // R/L=1 Desplazar a la derecha. R/L=0 izda.
byte ems;            // --- Entry mode set ---
                     // --- 0 0 0 0 0 1 I/D S
                     // I/D = 1: Incrementar contador de direcciones
                     // I/D = 0: Decrementar contador direcciones
                     // S = 1: Desplazar display, S = 0 Desp. cursor

/* ----- Caracteres definidos ----- */

byte llave[8] = {
  0x0E,
  0x11,
  0x0E,
  0x04,
  0x04,
  0x0C,
  0x04,
```

```
  0x0C
};

byte comecocos1[8] = {
  0x07,
  0x0B,
  0x0E,
  0x1C,
  0x18,
  0x1C,
  0x0E,
  0x07
};
byte comecocos2[8] = {
  0x00,
  0x06,
  0x0B,
  0x1F,
  0x1F,
  0x0F,
  0x06,
  0x00
};

/*
           +-----------------------------------------+
 ----------¦     RUTINAS DE GESTION DE LA CT6811.    ¦------------
           +-----------------------------------------+             */

void accion_rxcar()
{
}

void accion_break()
{
}

void carga()
{
  static byte n=0;
  static byte i=0;

  if (n==16 || i==255) {
    n=0;
    printf ("_");
  }

  i++;
  n++;
}

int inicializar_ct6811()
/*
+-------------------------------------------------------------------------+
¦  Incializar la CT6811 con el CTSERVER. Se comprueba si el CTSERVER      ¦
¦  está cargado. En caso de no estarlo se carga.                          ¦
¦                                                                         ¦
¦  La función devuelve 1 en caso de que todo funcione. 0 en caso de       ¦
¦  haberse establecido la conexion con el servidor.                       ¦
+-------------------------------------------------------------------------+*/

{
  char c;
  S19 fs19;
  char *caderror;

  abrir_puerto_serie(2);  /* Abrir sesion con puerto serie COM2 */
  baudios(9600);

  printf ("\n");
  check_conexion();
  if (!hay_conexion()) {
    printf ("NO HAY CONEXION\n");
    if (abrir_s19("ctserver.s19",&fs19,1)==0) {
      caderror=(char *)geterrors19();
      printf ("Error: %s\n",caderror);
      return 0;
    }
    printf ("\n");

    printf ("0%     50%   100%\n");
```

```
    printf ("_________________\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b");

    resetct6811();
    if (cargars19_ramint(fs19,carga)==0) {
      printf ("\nError: %s",getloaderror());
      cerrar_s19(fs19);
      return 0;
    }
    else {
      printf (" OK!!\n\n");
      cerrar_s19(fs19);
      delay(1000);
      baudios(9600);
    }
  }

  check_conexion();
  if (!hay_conexion()) {
    printf ("Conexion no establecida\n");
    cerrar_puerto_serie();
    return 0;
  }
  printf ("CONEXION ESTABLECIDA\n");
  return 1;
}



/*         +-------------------------------------------+
           ¦     RUTINAS DE NIVEL FISICO DEL LCD.      ¦
           ¦ La implementación de estas rutinas depende¦
           ¦ de la opción de conexionado.              ¦
           +-------------------------------------------+              */


/*--- INTERFAZ ---*/

void dato_lcd(byte dat);
void cmd_lcd(byte cod);
void init_lcd();

/*--- Rutinas de implementación --*/

void enviar_lcd(byte dat);


void init_lcd()
/*
+------------------------------+
¦ Inicializar el LCD a 8 bits. ¦
+------------------------------+*/
{
  store(0xFF,DDRC);   // Configurar PUERTO C para salida
  store(0x00,PORTC);

  enviar_lcd(0x30);   // Configurar para 8 bits
  enviar_lcd(0x80);   // Relleno...

  enviar_lcd(0x20);

  enviar_lcd(0x20);
  enviar_lcd(0x80);

  cmd_lcd(control_lcd);  // Encender display y cursor
  cmd_lcd(0x01);         // CLS

}

void cmd_lcd(byte cod)
/*
+------------------------------+
¦ Ejecutar un comando del LCD ¦
+------------------------------+*/
{
   puertoc=0x00;               // RS=0; R/W=0;
   enviar_lcd(cod&0xF0);       // Enviar 4 bits mas signfic. con RS=0, R/W=0
   enviar_lcd((cod<<4)&0xF0);  // Enviar 4 bits menos signif.
}

void dato_lcd(byte dat)
/*
```

```
+----------------------------+
¦  Enviar un carácter al LCD  ¦
+----------------------------+*/
{
  puertoc=0x02;             // RS=1
  enviar_lcd(dat&0xF0);
  enviar_lcd((dat<<4)&0xF0);
}

void enviar_lcd(byte dat)
{
  puertoc&=0x0F;
  puertoc|=(dat & 0xF0);
  store(puertoc,PORTC);
  puertoc|=0x01;
  store(puertoc,PORTC);   // E=1
  puertoc&=~0x01;
  store(0x00,PORTC);      // E=0
}


/*          +-----------------------------------------+
            ¦ RUTINAS DEL NIVEL DE APLICACION.....    ¦
            +-----------------------------------------+              */

void print_lcd(char *cad)
/*
+-------------------------------+
¦  Enviar una cadena al LCD.    ¦
+-------------------------------+*/
{
  byte dat;
  int i;

  i=0;

  while (cad[i]!=0) {
    dat=cad[i];
    dato_lcd(dat);
    i++;
  }
}

void define_car(int cod, byte *car)
/*
+-------------------------------------------------------------------------+
¦  Definir un caracter por el usuario. Cod indica el número de carácter¦
¦  Car es un array de 8 bytes que contiene los datos del nuevo caracter¦
+-------------------------------------------------------------------------+*/
{
  byte dir;
  int i;

  dir = (byte)(cod*8)&0xFF;
  dir = dir | 0x40;

  cmd_lcd(dir);
  for (i=0; i<8; i++) {
    dato_lcd(car[i]);
    printf (".");
  }
  cmd_lcd(0x80);
}

void locate(int x,int y)
/*
+-----------------------------------------------+
¦  Situar el cursor del LCD en la posición (x,y) ¦
+-----------------------------------------------+*/
{
  byte dir;

  dir=(x-1) + (y-1)*0x40;
  dir=dir | 0x80;
  cmd_lcd(dir);
}

void cls()
/*
+------------------+
¦  Borrar el LCD   ¦
```

```
+------------------+*/
{
  cmd_lcd(0x01);
}

void home()
/*
+------------------+
¦ Cursor a HOME    ¦
+------------------+*/
{
  cmd_lcd(0x02);
}

void estado_cursor()
/*
+---------------------------------+
¦ Activar/desactivar el cursor    ¦
+---------------------------------+*/
{
  control_lcd=control_lcd^0x02;
  cmd_lcd(control_lcd);
}

void estado_display()
/*
+---------------------------------+
¦ Activar/desactivar el display   ¦
+---------------------------------+*/
{
  control_lcd=control_lcd^0x04;
  cmd_lcd(control_lcd);
}

void estado_parpadeo()
/*
+---------------------------------+
¦ Activar/desactivar el parpadeo  ¦
+---------------------------------+*/
{
  control_lcd=control_lcd^0x01;
  cmd_lcd(control_lcd);
}

void lcd_left()
/*
+---------------------------------+
¦ Desplazar el LCD a la izqda.    ¦
+---------------------------------+*/
{
  shift_lcd=0x18;
  cmd_lcd(shift_lcd);
}

void lcd_right()
/*
+---------------------------------+
¦ Desplazar el LCD a la derecha   ¦
+---------------------------------+*/
{
  shift_lcd=0x1C;
  cmd_lcd(shift_lcd);
}

void cursor_left()
/*
+---------------------------------+
¦ Mover el cursor a la izqda.     ¦
+---------------------------------+*/
{
  shift_lcd=0x10;
  cmd_lcd(shift_lcd);
}

void cursor_right()
/*
+---------------------------------+
¦ Mover el cursor a la derecha.   ¦
+---------------------------------+*/
{
  shift_lcd=0x14;
```

```
  cmd_lcd(shift_lcd);
}

void prueba()
{
  ems^=0x01;
  cmd_lcd(ems);
}

void inicializar_lcd()
{
  control_lcd=0x0E;  // Por defecto Cursor ON, LCD on y no parpadeo
  shift_lcd=0x10;    // Por defecto desplazar cursor
  ems=0x06;
  puertoc=0;

  init_lcd();

  define_car(0,llave);      // Caracter 0 --> llave
  define_car(1,comecocos1); // Caracter 1 --> Comecocos abierto
  define_car(2,comecocos2); // Caracter 2 --> Comecocos cerrado
}


void presenta_lcd()
{
  cls();
  print_lcd(" *PROBANDO LCD*");
  locate(1,2);
  print_lcd("PROYECTO LOCK ");
  dato_lcd(0);                // Sacar la llave
  home();
}

void comecocos()
{
  byte estado=0;
  int i;

  home();
  control_lcd=0x0C;
  cmd_lcd(control_lcd);

  for(i=1; i<=16; i++) {
    locate(i,1);
    dato_lcd(' ');
    dato_lcd(estado+1);
    delay(100);
    estado^=0x01;
  }
  home();
  control_lcd=0x0E;
  cmd_lcd(control_lcd);
}


void lcd()
{
  char c;

  clrscr();
  printf ("    PRUEBA DEL LCD CONECTADO A 8 BITS\n");
  printf ("    --------------------------------\n\n");

  presenta_lcd();

  printf ("1.- CLS\n");
  printf ("2.- Presentacion\n");
  printf ("3.- Cursor a HOME\n");
  printf ("4.- Cursor ON/OFF\n");
  printf ("5.- Display ON/OFF\n");
  printf ("6.- Estado parpadeo\n");
  printf ("7.- Desplazar LCD a la izda\n");
  printf ("8.- Desplazar LCD a la der.\n");
  printf ("9.- Cursor izquierda\n");
  printf ("0.- Cursor derecha\n\n");
  printf ("ENTER.- Comecocos\n");
  printf ("ESC.- Terminar\n\n");
  printf ("Opcion: ");
```

```
  do {
    c=(char)toupper(getch());
    switch(c) {
      case '1': cls(); break;
      case '2': presenta_lcd(); break;
      case '3': home(); break;
      case '4': estado_cursor(); break;
      case '5': estado_display(); break;
      case '6': estado_parpadeo(); break;
      case '7': lcd_left(); break;
      case '8': lcd_right(); break;
      case '9': cursor_left(); break;
      case '0': cursor_right(); break;
      case '+': prueba(); break;
      case 27 : break;
      case 13 : comecocos(); break;
      default : dato_lcd(c);
    }
  } while (c!=27);

}

main()
{

  if (inicializar_ct6811()==0) {    /* Inicializar CT6811) */
    cerrar_puerto_serie();
    return 0;
  }
  delay(500);

  clrscr();
  printf ("DEFINIENDO CARACTERES ESPECIALES.");

  inicializar_lcd();    /* Inicializar LCD  */

  lcd();         /* Programa de control del LCD */

  cerrar_puerto_serie();     /* Cerrar la sesión con el COM2 */
  return 0;
}
```