Laboratorio de Estructura de Computadores
Curso 2007-08
- 2ª Parte -

Práctica 6

Manejo de la pantalla de cristal líquido (LCD)

6.1. Práctica 6 - Manejo del LCD

6.1.1. Objetivos de la práctica

- Familiarizarse con el manejo de la pantalla de cristal líquido (LCD, *Liquid Cristal Display*) y comprender su funcionamiento.
- Aprender a programar el controlador del LCD.
- Aprender a visualizar texto y gráficos sobre el LCD.

Para adquirir estos conocimientos se va a realizar un programa que dibuje caracteres ASCII y rectángulos en la pantalla LCD incluida en la placa de desarrollo. Pero antes de describir cómo implementar el código de la práctica se va a presentar una serie de conceptos necesarios para la comprensión de la misma.

6.1.2. Conceptos teóricos

Elementos hardware

La placa S3CEV40 contiene una pantalla de cristal líquido cuya finalidad es visualizar texto e información gráfica. La Tabla 6.1 presenta las características de dicha pantalla.

Modelo:	LRH9J515XA STN/BW
Tamaño:	320×240 píxeles
Tamaño pantalla (diagonal):	9.6 cm
Tamaño punto:	0.24 mm
Color:	graduación de 16 niveles de gris
Modo scan:	*4-bit single scan*
Alimentación:	21.5V (25 ℃)

Cuadro 6.1: Parámetros del LCD.

En el mercado existen dos tipos de pantallas LCD[1]: pantallas de matriz activa y pantallas de matriz pasiva. Entre las tecnologías de matriz pasiva se encuentra la tecnología STN (*Super-Twisted Nematic*) que es la tecnología empleada en el LCD de la placa S3C44B0X y que usaremos en esta práctica. Esta tecnología puede producir imágenes en dos modos: modo positivo y modo negativo. En el modo positivo proporciona un fondo blanco con segmentos negros o en escala de grises. En el modo negativo proporciona un fondo negro con segmentos blancos o en escala de grises. En cuanto a sus características más destacables, desde el punto de vista de su uso, este tipo de LCD posee tiempos de respuesta lentos y un contraste bastante limitado por lo cual tiene un coste de fabricación bajo, siendo especialmente indicado para aplicaciones que requieran pantallas monocromas pequeñas (pantallas calculadoras, relojes digitales, etc.).

El LCD está conectado a un circuito encargado de proporcionarle la alimentación y de realizar ciertas adaptaciones en las señales que recibe (adaptación de los niveles de tensión, amplificación de los valores de corriente, etc.). Este circuito recibe el nombre de *driver*.

[1] Desde ahora usaremos el término LCD para referirnos a la pantalla LCD

Por último, el circuito *driver* recibe sus entradas de un circuito controlador (*controller*) incluido dentro del S3C44B0X. El controlador se encarga de:

1. Realizar la transferencia de la imagen que se desea visualizar en el LCD desde una memoria de vídeo localizada en la memoria del sistema hasta el *driver* del LCD.
2. Generar las señales de control necesarias para el correcta visualización en el LCD.

Controlador del LCD

El controlador del LCD se encarga tanto de la transferencia de los datos de vídeo desde la memoria de vídeo (localizada en la memoria de sistema) hasta el propio controlador, como de la generación de las señales de datos de vídeo, VD[0:7], y control ,VFRAME, VLINE, VCLK y VM, que son enviadas desde el controlador al *driver*. La Figura 6.1 muestra el diagrama de bloques de este dispositivo. Consta de cuatro bloques (REGBANK, LCDCDMA, VIDPRCS y TIMEGEN) cuya función detallamos a continuación.

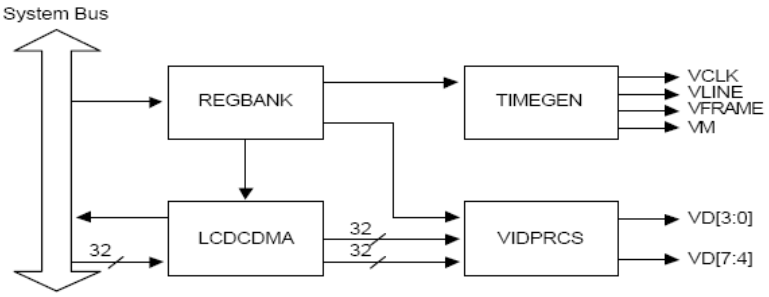

Figura 6.1: Diagrama de bloques del controlador.

- El bloque REGBANK contiene 18 registros programables a través de los cuales se realiza la configuración del controlador.
- El bloque LCDCDMA es un DMA (*Direct Access Memory*) encargado de transferir datos desde la memoria de vídeo al controlador sin intervención de la CPU, consiguiendo una transferencia de datos con mínimo retardo.
- El bloque VIDPRCS recibe los datos del bloque LCDCDMA y los envía al *driver* del LCD a través del puerto de datos VD[7:0], realizando previamente una conversión de formatos para adaptarlos a los requisitos del *driver*.
- Finalmente, el bloque TIMEGEN contiene lógica programable para generar las diferentes tasas y velocidades de transmisión que utilizan los distintos tipos de *drivers*.

Una vez visto el diagrama de bloques, estamos en condiciones de esbozar el flujo de datos en el controlador. Dentro del bloque LCDCDMA hay una memoria FIFO de 24 palabras. Cuando esta memoria FIFO está parcial o totalmente vacía, el bloque LCDCDMA solicita nuevos datos a la memoria de vídeo localizada en la memoria del sistema. En la Figura 6.1 se puede apreciar que el controlador está conectado al bus de sistema, luego para que la transferencia pueda ocurrir es necesario que dicha petición sea aceptada por el árbitro del

bus. Cuando ésto ocurre, cuatro palabras (16 bytes) son transmitidos desde la memoria de sistema a la memoria FIFO en el controlador. A continuación, la imagen se transfiere desde la memoria FIFO al *driver* del LCD usando el puerto VD[7:0].

Interfaz del controlador LCD

El controlador proporciona las señales de control VFRAME, VLINE, VCLK y VM que guían la transferencia de datos hacia el *driver*. Todas estas señales son generadas en el módulo TIMEGEN a partir de los valores escritos en los registros de configuración LCDCON1 y LCDCON2 que están contenidos en el bloque REGBANK (véase la sección "Registros del controlador" para leer una descripción más detallada de estos registros). La Figura 6.2 ilustra el funcionamiento de estas señales, cuya descripción se presenta a continuación con algo más detalle.

- **VCLK**: Esta línea transporta la señal de reloj entre el controlador de LCD y el *driver* del LCD. Los datos son enviados por el controlador del LCD en el flanco de subida de VCLK y registrados por el *driver* del LCD en el flanco de bajada de VCLK.

  El valor de la frecuencia de la señal VCLK se controla con el campo CLKVAL del registro de configuración LCDCON1. Dicho parámetro viene gobernado por la siguiente ecuación:

$$VCLK(Hz) = \frac{MCLK}{2 \cdot CLKVAL} \tag{6.1}$$

  Así, la Tabla 6.2 muestra la relación entre CLKVAL y la frecuencia de VCLK suponiendo un reloj principal a 60MHz.

CLKVAL	60MHz/X	VCLK
2	60MHz/4	15.0MHz
3	60MHz/6	10.0MHz
⋮	⋮	⋮
1023	60MHz/2046	29.3KHz

Cuadro 6.2: Relación entre VCLK y CLKVAL.

- **VFRAME**: señal de sincronía de marco (*frame*) entre el controlador y el *driver* del LCD. Esta señal indica el comienzo de un nuevo marco. En otras palabras, esta señal es activada para llevar el puntero de línea del LCD al comienzo del marco e iniciar el dibujo de una nueva pantalla. El controlador la activa después de que se haya completado la visualización de un marco en la pantalla, y, por lo tanto, su frecuencia de activación es de un pulso por marco. Por último, el pulso de la señal VFRAME tiene la duración de la primera línea de pantalla.

- **VLINE**: señal de sincronía de línea entre el controlador y el *driver*. Es utilizada por el *driver* del LCD para transferir los contenidos desde el registro de desplazamiento de línea horizontal a la pantalla. El registro de desplazamiento contenido dentro del

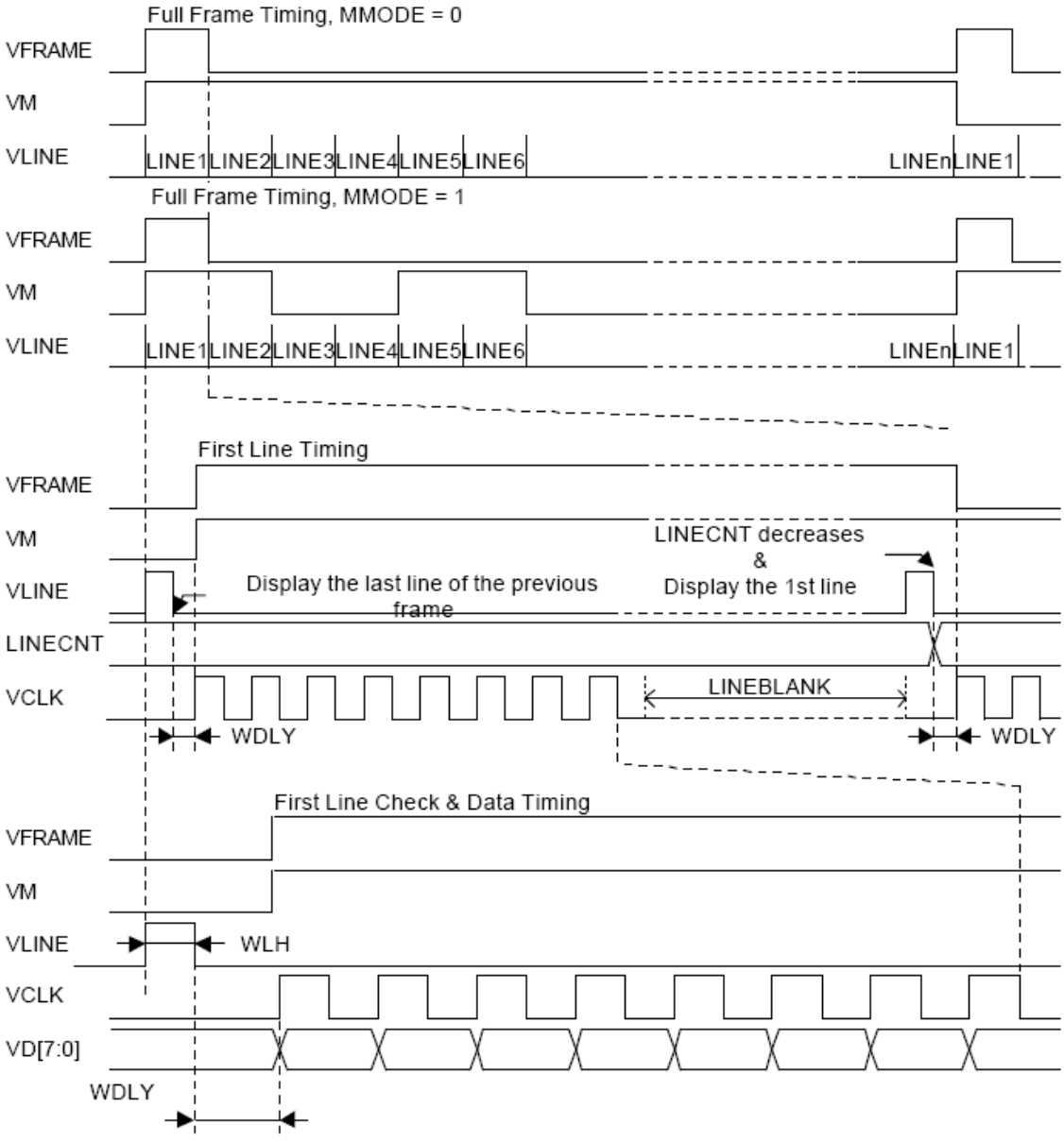

Figura 6.2: Diagramas de tiempo de las señales de control generadas por el controlador trabajando en modo 8-bit single scan.

driver utiliza la señal VCLK como reloj para sincronizar esta operación. El controlador activa la señal VLINE después de que una línea horizontal de datos haya sido completamente cargada en el registro de desplazamiento contenido en el *driver* del LCD.

Tal y como se ilustra en la Figura 6.2, el cronograma de las señales VFRAME y VLINE se establece en función de los parámetros WLH (anchura de pulsos de la señal VLINE), WHLY (retraso de VCLK después del pulso de VLINE), VLINEBLANK, HOZVAL y LINEVAL, todos ellos definidos como campos en los registros de configuración LCDCON1 y LCDCON2. Así mismo, el valor de los parámetros HOZVAL y LINEVAL se determina a partir del tamaño del LCD y el modo de visualización en el que este trabaja según la ecuación:

$$HOZVAL = \frac{Tamano\ horizontal\ pantalla}{Numero\ lineas\ datos\ validas\ en\ VD} - 1 \tag{6.2}$$

$$LINEVAL4 = Tamano\ vertical\ pantalla - 1 \tag{6.3}$$

$$LINEVAL8 = \frac{Tamano\ vertical\ pantalla}{2} - 1 \tag{6.4}$$

donde (6.3) es la ecuación a utilizar para calcular el valor del parámetro en modo *4-bit dual scan* y (6.4) es la ecuación a utilizar para calcular el valor del parámetro en modo *8-bit single scan*. Así mismo, en las anteriores ecuaciones, el número de líneas válidas en modo *4-bit dual scan* es 4 y en el modo *8-bit single scan* es 8. Véase la sección "Modos de operación" para encontrar una descripción detallada de los mismos.

- **VM**: Esta es la señal AC para el *driver* del LCD. Esta señal es utilizada por el *driver* para alternar la polaridad de la tensión de filas y columnas; tensión utilizada para encender y apagar los píxeles de la pantalla. Esta señal puede cambiar tras cada marco o tras un número de transiciones de la señal VLINE. La configuración de su modo de trabajo se realiza en los campos MMODE y MVAL del registro de configuración LCDCON1. Si MMODE= 0 la señal VM conmuta cada marco. Si MMODE= 1 la señal VM conmuta tras el número de transiciones de VLINE indicado por MVAL según la siguiente ecuación:

$$VM\ Rate = \frac{VLINE\ Rate}{2 \cdot MVAL} \tag{6.5}$$

  La Figura 6.3 muestra dos ejemplos de su funcionamiento. En el primero MMODE= 0 y en la segunda MMODE= 1 y MVAL[7:0]=0x1.

- **VD[7:0]**: Puerto de salida de datos.

Localización del marco en la memoria de sistema

Como ya se ha mencionado con anterioridad la memoria de vídeo tiene asignado un rango del espacio de direcciones de memoria y contiene la imagen que está siendo visuali-

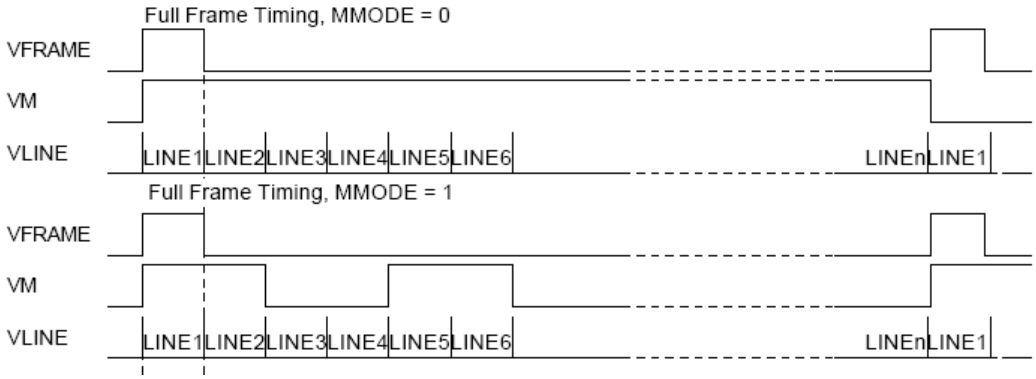

Figura 6.3: Diagrama de tiempos de la señal VM para dos posibles configuraciones: MMODE=0; MMODE=1 y MVAL=1.

zada en la pantalla. Los datos de vídeo son transferidos desde esta memoria a la memoria FIFO en el controlador usando el DMA que posee este dispositivo.

Existe otra memoria, a la que llamaremos memoria virtual de vídeo, que también tiene asignado un rango del espacio de direcciones de memoria y que contiene la siguiente imagen a ser visualizada en la pantalla. Para ver esta imagen en el LCD es necesario mover el contenido de la memoria virtual de vídeo a la memoria de vídeo y desde allí los datos son transportados al controlador del LCD mediante el DMA incluido en este dispositivo. El movimiento de los datos desde la memoria virtual de vídeo a la memoria de vídeo será realizado por uno de los canales de DMA definidos en el S3C44B0X y cuyo funcionamiento se verá con más detalle en la sección "DMA". En cualquier caso es necesario aclarar que en la transferencia de los datos desde la memoria virtual de vídeo hasta el controlador están involucrados dos DMA distintos: uno localizado en el S3C44B0x que mueve los datos dentro de la memoria de sistema desde la memoria virtual de vídeo a la memoria de vídeo; otro localizado en el controlador del LCD que transfiere los datos desde la memoria de vídeo a dicho controlador.

Finalmente, el tamaño de la memoria de vídeo no tiene porque ser igual al tamaño del LCD. En otras palabras, el tamaño de la memoria de vídeo puede ser mayor que el tamaño del LCD. En esa situación, el LCD solo visualiza una porción de la memoria de vídeo y es posible realizar desplazamientos (*scrolling*) tanto vertical como horizontales modificando la porción de la memoria que se visualiza. La Figura 6.4 ilustra el efecto de *scrolling*.

La porción de la memoria de vídeo que se visualiza viene controlada por cuatro parámetros: posición del borde superior (LCDBASEU), posición del borde inferior (LCDBASEL), ancho de pantalla (PAGEWIDTH) y offset respecto del borde de la pantalla (OFFSIZE). Estos cuatro parámetros se almacenan en los registros LCDSADDR1 y LCDSADDR2 del controlador de LCD.

Modos de operación

El controlador de LCD puede ser configurado para trabajar con tres tipos distintos de LCDs: LCDs que trabajen en modo 4-bit dual scan, LCDs que trabajen en modo 4-bit single scan o LCDs que operen en modo 8-bit single scan.

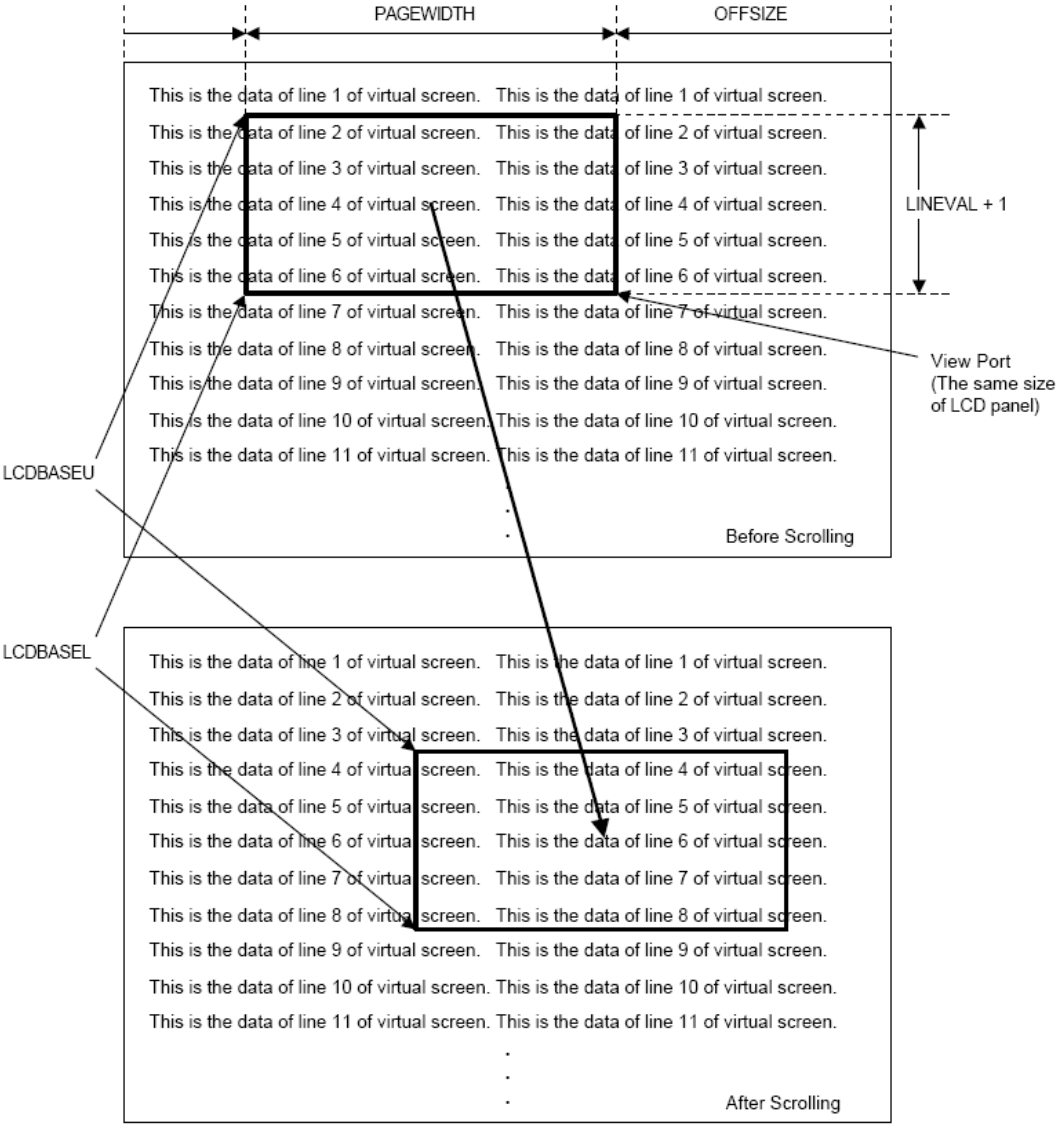

Figura 6.4: Ejemplo de *scrolling*.

- *4-bit dual scan*: Este tipo de LCD utiliza líneas de datos de 8 bits para dibujar simultáneamente en la mitad superior e inferior de la pantalla. Así, cuatro bits son utilizados para dibujar la parte superior de la pantalla y los otros cuatro bits son utilizados para dibujar la mitad inferior. El final de un marco se alcanza cuando cada mitad de la pantalla ha sido completamente dibujada. Con este tipo de LCD, los ocho pines del puerto VD[7:0] pueden ser conectados directamente al *driver* del LCD. La Figura 6.5 ilustra este modo de operación.

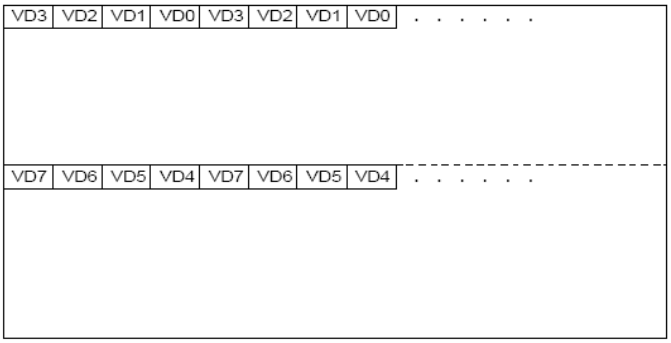

Figura 6.5: Pantalla en modo *4-bit dual scan*.

- *4-bit single scan*: Este tipo de LCD utiliza líneas de datos de 4 bits para dibujar cada una de las líneas horizontales de la pantalla, una línea cada vez. Con este tipo de LCD, cuatro pines del puerto VD, VD[3:0], deben estar conectados al *driver* y los cuatro pines restantes, VD[7:4], no son utilizados. La Figura 6.6 ilustra este modo de operación.

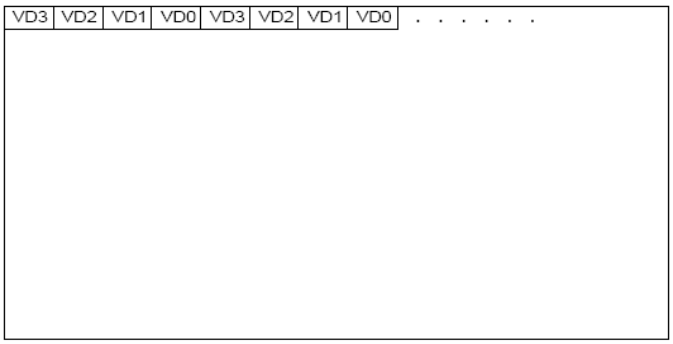

Figura 6.6: Pantalla en modo *4-bit single scan*.

- *8-bit single scan*: Este tipo de LCD utiliza líneas de datos de 8 bits para dibujar cada una de las líneas horizontales de la pantalla, una línea cada vez. Con este tipo de LCD, los ocho pines del puerto VD[7:0] estarán conectados directamente al *driver* del LCD. La Figura 6.7 ilustra este modo de operación.

El modo de scan se selecciona con el campo DISMOD del registro de configuración LCDCON1.

Figura 6.7: Pantalla en modo *8-bit single scan*.

Registros del controlador

El controlador del LCD puede estar conectado a diferentes tipos de *drivers* o, lo que es lo mismo, a diferentes tipos de LCDs. Por lo tanto debe ser programable para poder adaptarse a los diferentes requisitos de los distintos tipos de pantalla: número de líneas horizontales y verticales, ancho de la línea de datos, etc. En particular el controlador del LCD incluido en la placa de desarrollo es capaz de funcionar con tres tipos de pantallas LCD: monocromo, escala de grises y color, si bien la pantalla incluida en el entorno de desarrollo es una pantalla LCD de escala de grises con 16 niveles de gris (véase Tabla 6.1).

La configuración del controlador se realiza programando los registros que este circuito con los valores necesarios para adaptarse a los requisitos del LCD. Dedicaremos las siguientes líneas a presentar los registros de configuración que van a ser utilizados en esta práctica. Para ver todos los registros involucrados en la configuración del LCD consúltese el capítulo 12 del manual de la placa S3C44B0X.

- LCD Control 1 Register (LCDCON1)
  - Frecuencia VCLK.
  - Formas de onda de las señales VFRAME, VLINE y VM.
  - Modo de scan.
  - Polaridad de las señales de control.
  - Habilitar lógica del controlador/Limpiar memoria FIFO.

  La Figura 6.8 muestra con más detalle los campos de este registro.
- LCD Control 2 Register (LCDCON2)
  - Formas de onda de las señales VFRAME y VLINE.
  - Tamaño del LCD.

  La Figura 6.9 muestra con más detalle los campos de este registro.
- Frame Buffer Start Address 1 Register (LCDSADDR1)
  - Modo de trabajo.

Register	Address	R/W	Description	Reset Value
LCDCON1	0x01F00000	R/W	LCD control 1 register	0x00000000

LCDCON1	Bit	Description	Initial State
LINECNT (read only)	[31:22]	These bits provide the status of the line counter. Down count from LINEVAL to 0	0000000000
CLKVAL	[21:12]	These bits determine the rate of VCLK. If this value can be changed when ENVID=1, the new value will be used next frame. *VCLK = MCLK / (CLKVAL x 2) (CLKVAL ≥ 2)*	0000000000
WLH	[11:10]	These bits determine the VLINE pulse's high level width by counting the number of the system clock. 00 = 4 clock, 01 = 8 clock, 10 = 12 clock, 11 = 16 clock	00
WDLY	[9:8]	These bits determine the delay between VLINE and VCLK by counting the number of the system clock 00 = 4clock, 01 = 8 clock, 10 = 12 clock, 11 = 16 clock	00
MMODE	[7]	This bit determines the toggle rate of the VM. 0 = Each Frame, 1 = The rate defined by the MVAL	0
DISMODE	[6:5]	These bits select the display mode. 00 = 4-bit dual scan display mode 01 = 4-bit single scan display mode 10 = 8-bit single scan display mode 11 = Not used	00
INVCLK	[4]	This bit controls the polarity of the VCLK active edge. 0 = The video data is fetched at VCLK falling edge 1 = The video data is fetched at VCLK rising edge	0
INVLINE	[3]	This bit indicates the line pulse polarity. 0 = normal 1 = inverted	0
INVFRAME	[2]	This bit indicates the frame pulse polarity. 0 = normal 1 = inverted	0
INVVD	[1]	This bit indicates the video data(VD[7:0]) polarity. 0 = Normal 1 = VD[7:0] output is inverted.	0
ENVID	[0]	LCD video output and the logic enable/disable. 0 = Disable the video output and the logic. The LCD FIFO is cleared. 1 = Enable the video output and the logic.	0

Figura 6.8: Campos del registro LCDCON1.

Register	Address	R/W	Description	Reset Value
LCDCON2	0x01F00004	R/W	LCD control 2 register	0x00000000

LCDCON2	Bit	Description	Initial State
LINEBLANK	[31:21]	These bits indicate the blank time in one horizontal line duration time. These bits adjust the rate of the VLINE finely. The unit of LINEBLANK is MCLK. Ex) If the value of LINEBLANK is 10, the blank time is inserted to VCLK during 10 system clocks.	0x000
HOZVAL	[20:10]	These bits determine the horizontal size of the LCD panel. HOZVAL has to be determined to meet the condition that total bytes of 1 line be 2n bytes. If the x size of LCD is 120 dots in mono mode, x=120 can not be supported because 1 line consists of 15 bytes. Instead, x=128 in mono mode can be supported because 1 line consists of 16 bytes(2n). The additional 8 dot will be discarded by LCD panel driver.	0x000
LINEVAL	[9:0]	These bits determine the vertical size of LCD panel.	0x000

Figura 6.9: Campos del registro LCDCON2.

- Localización del marco de vídeo (*frame*) en la memoria de vídeo.

La Figura 6.10 muestra con más detalle los campos de este registro.

Register	Address	R/W	Description	Reset Value
LCDSADDR1	0x01F00008	R/W	Frame buffer start address 1 register	0x000000

LCDSADDR1	Bit	Description	Initial State
MODESEL	[28:27]	These bits select the monochrome, gray, or color mode. 00 = monochrome mode 01 = 4-level gray mode 10 = 16-level gray mode 11 = color mode	00
LCDBANK	[26:21]	These bits indicate A[27:22] of the bank location for the video buffer in the system memory. LCDBANK value can not be changed even when moving the view port. LCD frame buffer should be inside aligned 4MB region, which ensures that LCDBANK value should not be changed when moving the view port. So, using the malloc function the care should be taken.	0x00
LCDBASEU	[20:0]	These bits indicate A[21:1] of the start address of the upper address counter, which is for the upper frame memory of dual scan LCD or the frame memory of single scan LCD.	0x000000

Figura 6.10: Campos del registro LCDSADDR1.

- Frame Buffer Start Address 2 Register (LCDSADDR2)
  - Parámetros de configuración de la señal de control VM.
  - Localización del marco de vídeo (*frame*) en la memoria de vídeo.

La Figura 6.11 muestra con más detalle los campos de este registro.

Valores de configuración del controlador

Tal y como se muestra en la Tabla 6.1 la pantalla LCD tiene un tamaño de 320×240 píxeles, trabaja en modo *4-bit single* y con una escala de 16 grises. La Tabla 6.3 muestra los valores de los parámetros del controlador que se deben utilizar para adaptarse a estas características.

LINEVAL	=	$(240\ -1)\|_{10}$	=	0xEF
PAGEWIDTH	=	$(320\ \times 4/16)\|_{10}$	=	0x50
OFFSIZE	=	512	=	0x200
LCDBANK	=	0xC300000 >> 22	=	0x30
LCDBASEU	=	0x1000000 >> 1	=	0x8000
LCDBASEL	=	0x8000 + (0x50+0x200) × (0xEF +1)	=	0xA2B00

Cuadro 6.3: Valores de los parámetros del controlador.

DMA

La transferencia de la imagen desde la memoria virtual de vídeo hasta la memoria de vídeo se realizará mediante DMA (*Direct Memory Access*) que permite transferir datos

Register	Address	R/W	Description	Reset Value
LCDSADDR2	0x01F0000C	R/W	Frame buffer start address 2 register	0x000000

LCDSADDR2	Bit	Description	Initial State
BSWP	[29]	Byte swap control bit 1 : Swap Enable 0 : Swap Disable LCD DMA fetches the frame memory data by 4 word burst access. In little endian mode and BSWP is 0, the frame memory data are displayed in the sequence, 4n+3th, 4n+2th ,4n+1th ,4n-th data. If BSWP is 1, the sequence will be 4n-th, 4n+1th, 4n+2th, 4n+3th. If the CPU is little endian mode, the frame buffer may be accessed by only byte access mode, Because BSWP is 1, the byte accessed data will be shown correctly also in the little endian mode. In the other case, BSWP has to be 0.	0
MVAL	[28:21]	These bits define the rate at which the VM signal will toggle if the MMODE bit is set to logic '1'.	0x00
LCDBASEL	[20:0]	These bits indicate A[21:1] of the start address of the lower address counter, which is used for the lower frame memory of dual scan LCD. LCDBASEL = LCDBASEU + (PAGEWIDTH + OFFSIZE) x (LINEVAL +1)	0x0000

NOTE: Users can change the LCDBASEU and LCDBASEL values for scrolling while LCD controller is turned on. But, users must not change the LCDBASEU and LCDBASEL registers at the end of FRAME by referring to the LINECNT field in LCDCON1 register. Because of the LCD FIFO fetches the next frame data prior to the change in the frame. So, if you change the frame, the pre-fetched FIFO data will be obsolete and LCD controller will display the incorrect screen. To check the LINECNT, interrutpt should be masked. If any interrupt is executed just after reading LINECNT, the read LINECNT value may be obsolete because of the execution time of ISR(interrupt service routine).

Figura 6.11: Campos del registro LCDSADDR2.

entre dos localizaciones sin intervención de la CPU, de manera que el sistema no se queda bloqueado realizando operaciones de E/S.

El S3C44B0X tiene cuatro canales de DMA: dos conectados al bus de sistema, ZDMA0 y ZDMA1, y dos localizados en el bridge que hacen de interfaz con el bus periférico, BDMA0 y BSMA1. Los dos controladores conectados al bus de sistema realizan la transferencia de datos desde memoria a memoria, o entre la memoria y dispositivos de E/S mapeados en memoria. Por lo tanto, en esta práctica vamos a utilizar uno de los controladores DMA conectados al bus de sistema porque estamos interesados en realizar el movimiento de datos entre las dos memorias de vídeo localizadas en la memoria de sistema. En particular, utilizaremos el controlador ZDMA0.

La Figura 6.12 muestra el diagrama del controlador ZDMA. Las señales nXDREQ y nXDACK (la letra n delante del nombre significa que estas señales son activas a nivel bajo) son las utilizadas para negociar con el árbitro del bus la petición y concesión del bus. En particular la señal nXDREQ es utilizada para solicitar el uso del bus y la señal nXDACK es utilizada por el árbitro del bus para realizar la concesión. La activación de la señal nXDREQ ocurre cuando comienza la operación del DMA y dicho comienzo puede activarse bien por HW bien por SW. En particular, en esta práctica el comienzo de la operación de DMA se realizará mediante SW. Con la finalidad de mejorar la utilización del bus y la velocidad de transferencia, dentro del ZDMA hay una memoria FIFO de 4 palabras que permite que las operaciones de transferencia se realicen en modo ráfaga (*burst*). Así, por ejemplo, durante la operación de DMA entre memorias es posible leer 4 palabras en

modo ráfaga y a continuación escribir estas 4 palabras, mejorando de esta forma el nivel de utilización del bus.

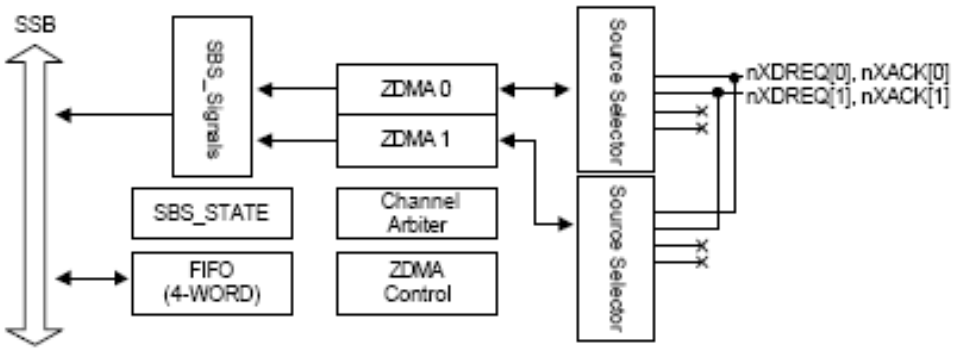

Figura 6.12: Diagrama de bloques del controlador ZDMA.

El DMA admite diferente número de operaciones de lectura/escritura por transferencia. La siguiente tabla muestra los tres modos de transferencia admitidos: Modo transferencia unidad (*Unit Transfer Mode*), modo transferencia bloque (*Block Transfer Mode*) y modo transferencia *on-the-fly* (*On-the-fly Transfer Mode*).

Modo de transferencia	Lectura/Escritura
Transferencia unidad:	lectura 1 unidad (palabra) y después escritura 1 unidad
Transferencia bloque:	lectura 4 unidades (palabras) y después escritura 4 unidades
Transferencia on-the fly:	o lectura 1 unidad o escritura 1 unidad

En esta práctica utilizaremos el modo de transferencia en bloque. En ese modo las transferencias se realizan en bloques de 4 palabras. Es decir, que hay cuatro ciclos de lectura, uno por palabra, antes de que se produzcan los cuatro ciclos de escritura, y que todos ellos se realizan sin solución de continuidad una vez ha comenzado la transferencia. En este modo de transferencia el tamaño total de los datos a ser transferidos debe ser múltiplo de 16 bytes (4 palabras) puesto que la unidad de cuenta del DMA es el byte. Si esto no fuera así, las n últimas palabras ($n < 4$) no se transferirían.

El DMA admite cuatro protocolos de comunicación, cada uno de los cuales define de forma distinta cómo se activan/desactivan las señales de petición (*request*) y reconocimiento (*acknowledge*) del protocolo. Estos modos son: *Handshake mode*, *single step mode*, *whole service mode* y *demand mode*. En esta práctica utilizaremos el modo *whole service*.

En el modo *whole service* las operaciones de DMA, cuya cantidad viene especificado por el número de unidades de transferencias definidas que se han de enviar, se iniciarán con una sola activación de la señal de petición (*request*) del DMA y la transferencia se completará sin nuevas peticiones del dispositivo DMA. En el caso de que el número de operaciones de DMA sea muy grande, como ocurre en esta práctica, el bus de sistema permanecerá ocupado durante el tiempo que dure la transferencia. En el caso de que el árbitro de bus asigne el uso del bus a un dispositivo más prioritario, el DMA interrumpirá la transferencia y una vez se le vuelva a asignar el bus la continuará en el punto en el que se encontraba.

El comienzo de la transferencia puede señalizarse mediante software o hardware. En esta

práctica se señalizará mediante software al escribir el valor 01 en el campo CMD del registro de configuración ZDCON0 y antes de iniciar la transferencia mediante DMA es necesario que todos los parámetros estén correctamente definidos: dirección de la fuente, dirección del destino, tamaño de la transferencia, etc. El final de la transferencias mediante DMA puede detectarse de dos formas distintas: mediante encuesta de la CPU al controlador DMA o mediante interrupción. La opción escogida en esta práctica es mediante interrupción; interrupción que se generará una vez se haya completado la transferencia de datos desde la memoria virtual de vídeo a la memoria de vídeo. En esta práctica no vamos a realizar ninguna tarea una vez finalizada la transferencia, luego lo único que deberá hacer la rutina de tratamiento de interrupción es borrar todas las interrupciones pendientes y activar un flag (`ucZdma0Done`) para indicar el final de la transferencia. La configuración de todos estos parámetros se realiza usando los registros de configuración del DMA, a saber: ZDCON0, ZDISRC0, ZDIDES0, ZDICNT0. Las Figuras 6.13, 6.14, 6.15, 6.16 muestran el significado de los campos de cada uno de estos registros.

- ZDMA Control Register (ZDCONn)
  - Almacena el estatus del canal DMA.
  - Habilita/deshabilita la solicitud externa de DMA
  - Control del inicio software de la transferencia DMA.

Register	Address	R/W	Description	Reset Value
ZDCON0	0x01E80000	R/W	ZDMA 0 Control Register	0x00
ZDCON1	0x01E80020	R/W	ZDMA 1 Control Register	0x00

ZDCONn	Bit	Description	Initial State
INT	[7:6]	Reserved	00
STE	[5:4]	Status of DMA channel (Read only) 00 = Ready 01 = Not TC yet 10 = Terminal Count 11 = N/A Before the DMA counter decreases from the initial counter value, STE is still in the ready state.	00
QDS	[3:2]	Disable/Enable External DMA request (nXDREQ) 00 = Enable other = Disable	00
CMD	[1:0]	Software commands 00: No command. After writing 01,10,11, CMD bit is cleared automatically. nXDREQ is available. 01: Starts DMA operation by S/W without nXDREQ. S/W start function can be used only in the whole mode. As DMA is in the whole mode, the DMA will operate until the counter is 0. If nXDREQ is used , this command must not be issued. 10: Pauses DMA operation. But nXDREQ is still available. 11: Cancels DMA operation.	00

NOTE: If users start the ZDMA operation by CMD=01b, the DREQ protocol must be whole service mode.

Figura 6.13: Campos del registro de configuración ZDCON.

- Initial Source Address (ZDISRCn)
  - Tamaño de la unidad de transferencia.
  - Dirección fuente de la transferencia.

ZDISRCn/ZDCSRCn	Bit	Description	Initial State
DST	[31:30]	Data size for transfer 00 = Byte, 01 = Half word 10 = Word, 11 = Not used If the block transfer mode is used, the DST must be 10.	00
DAL	[29:28]	Direction of address for load 00 = N/A, 01 = Increment 10 = Decrement, 11 = Fixed	00
ISADDR/CSADDR	[27:0]	Initial/current source address for ZDMAn	0x0000000

Figura 6.14: Campos del registro de configuración ZDISRC.

- Initial Destination Address Register (ZDIDESn)
  - Opciones internas del DMA.
  - Dirección destino de la transferencia.

ZDIDESn/ZDCDESn	Bit	Description	Initial State
OPT	[31:30]	DMA internal options. OPT = 10 is recommended. bit 31: Indicates how nXDREQ is sampled in the single step mode. 1 is recommended. bit 30: If the DST is half-word or word and if the DMA mode is not the block transfer mode, this bit takes a role. 1: DMA does word-swap or half-word swap Before transfer: B0,B1,B2,B3,B4,B5,B6,B7... word-swapped data: B3,B2,B1,B0,B7,B6,B5,B4,... half-word-swapped data: B1,B0,B3,B2,B5,B4,B7,B6,... 0: normal	00
DAS	[29:28]	Direction of address for store 00 = N/A 01 = Increment 10 = Decrement 11 = Fixed	00
IDADDR/CDADDR	[27:0]	Initial/current destination address for ZDMAn	0x0000000

Figura 6.15: Campos del registro de configuración ZDIDES.

- Initial Count Register (ZDICNTn)
  - Establece la señal que realiza la petición de DMA.
  - Protocolo de request.
  - Modo de transferencia.
  - Modo de interrupción.
  - Contador de la transferencia.

Además, cada uno de estos registros tiene asociado un registro de estado que contendrá los valores actuales , y no los valores iniciales, de cada uno de los campos. Los nombres de estos registros son: ZDCSRC0, ZDCDES0 y ZDCCNT (véase Figura 6.17). Estos registros son de solo lectura.

6.1.3. Desarrollo de la práctica

La práctica consiste en el dibujo sobre la pantalla de un mensaje en caracteres ASCII y de unos rectángulos. En particular se debe escribir el siguiente mensaje:

ZDICNTn/ZDCCNTn	Bit	Description	Initial State
QSC	[31:30]	DREQ(DMA request) source selection 00 = nXDREQ[0] 01 = nXDREQ[1] 10 = N/A 11 = N/A	00
QTY	[29:28]	DREQ protocol 00 = Handshake 01 = Single step 10 = Whole Service 11 = Demand	00
TMD	[27:26]	Transfer mode 00 = Not used 01 = Unit transfer mode 10 = Block(4-word) transfer mode 11 = On the fly If block transfer mode is selected, the ADDR[3:0] should be '0' to meet 16-byte align condition.	00
OTF	[25:24]	On the fly mode 00 = N/A 01 = N/A 10 = Read time on the fly 11 = Write time on the fly	00
INTS	[23:22]	Interrupt mode set 00 = Polling mode 01 = N/A 10 = Int. whenever transferred 11 = Int. whenever terminated count	00
AR	[21]	Auto-reload and Auto-start after DMA count are 0. 0 = Disable 1 = Enable. Even after DMA count is 0, the DMA H/W enable bit (EN bit) is still 1. But, DMA will start to operate only if the start command or nXDREQ is activated.	0
EN	[20]	DMA H/W enable/disable 0 = Disable DMA 1 = Enable DMA. If the QDS bit is 00b, DMA request can be serviced. Also if the S/W command is started, the DMA operation will occur. If the EN bit is 0, DMA will not operate even though S/W command is started. If the S/W command is canceled, the DMA operation will be canceled and EN bit will be cleared to 0. At the terminal count, the EN bit will be cleared to 0. **NOTE:** Do not set the EN bit and the other bits of ZDICNT register at the same time. User have to set EN bit after setting the other bits of ZDICNT register as following steps, 1. Set ZDICNT register with disabled En bit. 2. Set EN bit enable.	0
ICNT/CCNT	[19:0]	Initial/current transfer count for ZDMAn. If 1 byte is transferred, the ICNT will be decreased by 1. If 1 half-word is transferred, the ICNT will be decreased by 2. If 1 word is transferred, the ICNT will be decreased by 4. For example, if the data size of a transfer is word and the count is 4n+3, the last 3 bytes will not be transferred.	0x00000

Figura 6.16: Campos del registro de configuración ZDICNT.

Register	Address	R/W	Description	Reset Value
ZDCSRC0	0x01E80010	R	ZDMA 0 current source address Register	0x00000000
ZDCDES0	0x01E80014	R	ZDMA 0 current destination address Register	0x00000000
ZDCCNT0	0x01E80018	R	ZDMA 0 current count register	0x00000000

Figura 6.17: Descripción de los registros de estado del DMA.

`Código realizado por el puesto: <número de grupo>`
`Del grupo: 2B`

y debajo dibujar trece rectángulos con vértices y colores definidos por las siguientes n-tuplas:

(10,40,310,230,14), (20,45,300,225,13), (30,50,290,220,12), (40,55,280,215,11),
(50,60,270,210,10), (60,65,260,205,9), (70,70,250,200,8), (80,75,240,195,7),
(90,80,230,190,6), (100,85,220,185,5), (110,90,210,180,4), (120,95,200,175,3),
(130,100,190,170,2)

Cada componente de la n-tupla (x, y, u, v, c) tiene el siguiente significado: x coordenada x de la esquina superior izquierda del rectángulo; y coordenada y de la esquina superior izquierda del rectángulo; u coordenada x de la esquina inferior derecha del rectángulo; v coordenada y de la esquina inferior derecha del rectángulo; y finalmente, c color del rectángulo.

Como el elemento básico de dibujo es el píxel, cualquier imagen que quiera visualizarse sobre la pantalla deberá hacerse controlando y definiendo el valor de cada uno de los píxeles que la constituyen. Que se dibujen caracteres o rectángulos tan solo depende de cómo se definen los valores que deben tomar cada uno de los píxeles de la pantalla. La función `LCD_PutPixel(x, y, c)`, definida en el fichero `lcd.h`, es la función básica para dibujar píxeles en la pantalla; pinta el píxel de coordenadas x e y con el color c en la memoría virtual de vídeo. Las dos funciones que se han de definir en esta práctica, una para el dibujo de caracteres (`Lcd_DspAscII8x16`) y otra para el dibujo rectángulos (`Lcd_Draw_Box`), hacen uso de esta función.

Los caracteres ASCII que se van a dibujar tendrán un tamaño de 8×16 píxeles (véase Figura 6.18) y es necesario tener una descripción de los mismos en formato píxel. La tabla `const INT8U g_auc_Ascii8x16[]` definida en el fichero `AscII8x16.c` contiene esta descripción. Esta tabla es un array indexado por el valor ASCII de los caracteres. Cada fila del array consta de 16 valores, uno por cada una de las 16 líneas que constituyen el carácter, cada uno de los cuales es la codificación en hexadecimal de los valores de los píxeles en la línea: 1 si el píxel está encendido, 0 si el píxel está apagado. Por ejemplo, si el cuarto valor de una fila del array es 0xAF entonces los píxeles de la cuarta fila del carácter estarán en el estado que se muestra en la Figura 6.18.

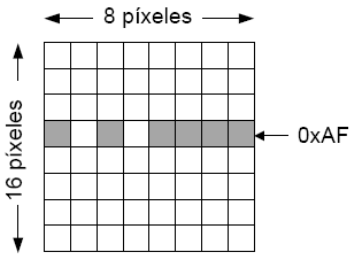

Figura 6.18: Ejemplo codificación de caracteres ASCII de tamaño 8x16 píxeles en el array `const INT8U g_auc_Ascii8x16[]`.

Diagrama de flujo

La figura 6.19 muestra tanto el diagrama de flujo del programa que vamos a realizar.

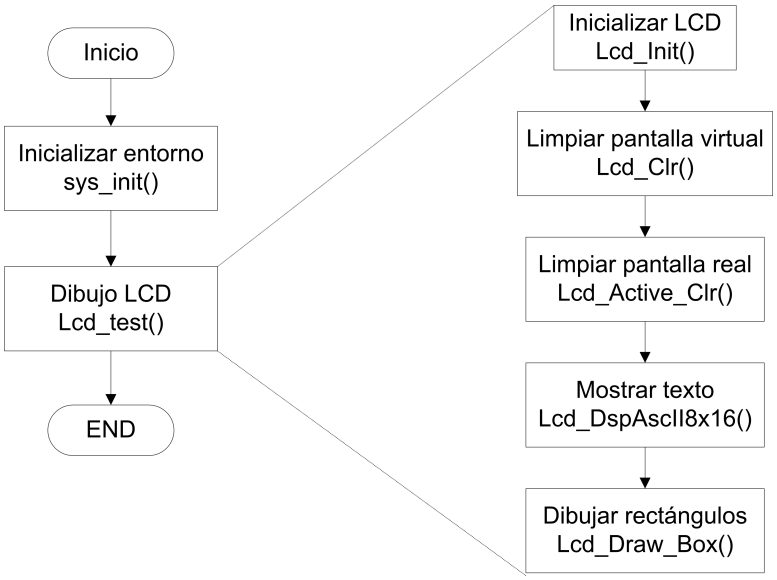

Figura 6.19: Diagrama de flujo.

Tal y como se muestra en la figura los pasos a seguir son:

1. Inicialización del entorno. Se realizará con la rutina `sys_init()` que ya vimos en prácticas anteriores.
2. Ejecución de la función `Lcd_test` que contiene todas las llamadas a las funciones que implementan la práctica:
   - `void Lcd_Init(void)`. Esta función configura el controlador para funcionar con el LCD incluido en la placa de desarrollo. La configuración se realiza a través de los registros de configuración incluidos dentro del controlador.
   - `void Lcd_Clr(void)`. Esta función borra los contenidos de la memoria virtual de vídeo. En otras palabras, pone al valor blanco todas las posiciones de memoria dentro del rango de direcciones de la memoria de sistema asignada a dicha memoria.
   - `void Lcd_Active_Clr(void)`. Esta función borra la memoria de vídeo. En otras palabras, pone al valor blanco en todos los píxeles de la pantalla.
   - `void Lcd_DspAscII8x16(INT16U x0, INT16U y0, INT8U ForeColor, INT8U * s)`. Esta función escribe en las posiciones de pantalla que se pasan por parámetro, x0,y0, el texto que también se pasa por parámetro, *s,. Para ello irá recorriendo cada uno de los caracteres del texto y dibujará cada carácter según el patrón indicado por el array `INT8U g_auc_Ascii8x16[]`, tal y como se muestra en la Figura 6.20.
   - `void Lcd_Dma_tran()`. Esta función configura e inicia la transferencia DMA entre la memoria virtual de vídeo y la memoria de vídeo.

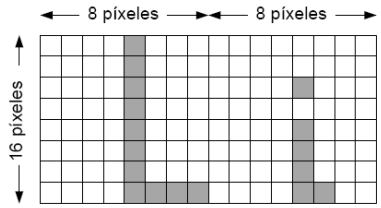

Figura 6.20: Ejemplo dibujo de los caracteres L e i sobre la pantalla.

- `void Lcd_Draw_Box(INT16 usLeft, INT16 usTop, INT16 usRight, INT16 usBottom, INT8U ucColor)`. Esta función dibuja un rectángulo con la forma y el color que se indican en los parámetros que se le pasan.

```
void Lcd_Draw_Box(INT16 usLeft, INT16 usTop, INT16 usRight,
                  INT16 usBottom, INT8U ucColor)
{
Lcd_Draw_HLine(usLeft, usRight,  usTop,    ucColor, 1);
Lcd_Draw_HLine(usLeft, usRight,  usBottom, ucColor, 1);
Lcd_Draw_VLine(usTop,  usBottom, usLeft,   ucColor, 1);
Lcd_Draw_VLine(usTop,  usBottom, usRight,  ucColor, 1);
}
```

  Como se aprecia en el código anterior, la función `Lcd_Draw_Box` llama a las funciones:

  - `void Lcd_Draw_HLine(INT16 usX0, INT16 usX1, INT16 usY0, INT8U ucColor, INT16U usWidth)`
  - `void Lcd_Draw_VLine (INT16 usY0, INT16 usY1, INT16 usX0, INT8U ucColor, INT16U usWidth)`

  que dibujan líneas horizontales y verticales desde las coordenadas X0,Y0 hasta las coordenadas X1,Y1 en el color ucColor y con un ancho de línea usWidth.

Organización del código de la práctica

La práctica consta de los siguientes ficheros:

1. Ficheros de código fuente:
   - `AscII8x16.c`: contiene la tabla `g_auc_Ascii8x16`.
   - `Bmp.c`: contiene funciones para el dibujo del cursor y de mapas de bits.
   - `lcd.c`: Contiene las funciones a desarrollar en la práctica.
   - `main.c`: Programa principal. Inicializa el entorno e invoca a la función que realiza el test del LCD.
2. Ficheros de cabecera:
   - `44b.h`, `44blib.h`, `def.h`, `option.h`: ya definidos en prácticas anteriores.

- `BMP.H`: definición de los prototipos de las funciones para dibujar el cursor y los mapas de bits.
- `lcd.h`: contiene directivas que definen los parámetros del LCD

3. Ficheros comunes: `44binit.s`, `44blib.c`, `ev40boot.cs`, `ram_ice.ld`, `uhal.c`: cuyas funcionalidades fueron descritas en prácticas anteriores.