Módulo LCD_CTRL

1. Funcionalidad

La finalidad de este módulo es la implementación de las operaciones básicas sobre la pantalla:

- Colocar el cursor en la posición X, Y;
- Dibujar uno (o más) píxeles desde la posición actual.

El esquema de entradas y salidas del módulo es el siguiente:

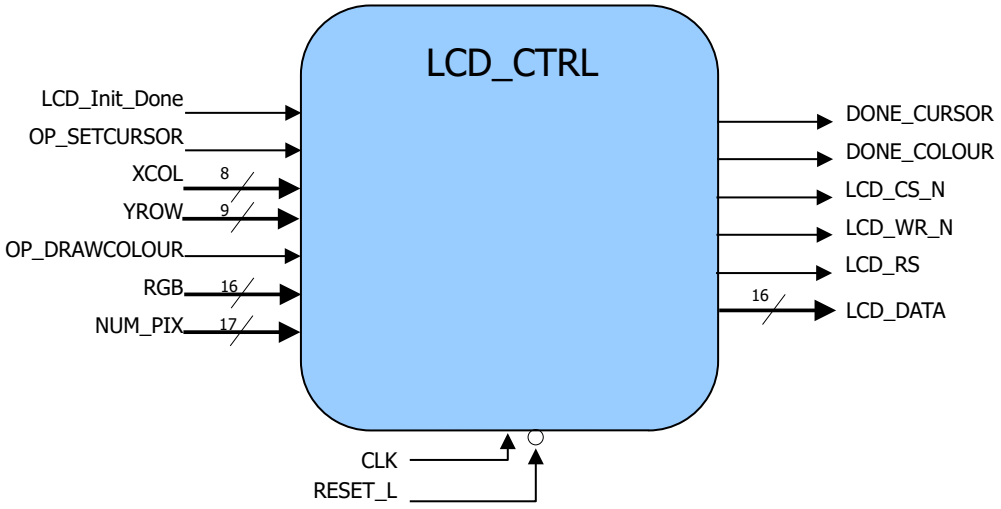

Figura 1

ENTRADAS:

- LCD_Init_Done: si su valor es 0, indica que la LCD está todavía en proceso de inicialización (el módulo LT24Setup controla esta señal) y no se puede realizar ninguna tarea. Cuando su valor es 1, el módulo LCD_CTRL podrá realizar las operaciones básicas.
- OP_SETCURSOR, XCOL e YROW: las tres entradas están relacionadas, la primera indica que se quiere realizar la operación de posicionamiento del cursor, y las dos siguientes indican la posición concreta. El valor de XCOL estará en el rango 0-239 (8 bits) y el de YROW entre 0-319 (9 bits).
- OP_DRAWCOLOUR, RGB y NUM_PIX: las tres entradas están relacionadas, la primera indica que se quiere realizar la operación de colorear pixels, y las dos siguientes indican el color (entrada RGB) y el número de pixels a colorear (entrada NUM_PIX).

SALIDAS

- DONE_CURSOR: indica que ha finalizado la operación de posicionamiento del cursor.
- DONE_COLOUR: indica que ha finalizado la operación de dibujar pixels.
- LCD_CS_N, LCD_WR_N, LCD_RS y LCD_DATA: tanto para realizar la operación de posicionamiento como para la de dibujar, hay que enviar una secuencia

concreta de información por esas salidas. Los detalles de esa secuencia están en la guía rápida de la pantalla LT24, y la información recogida en los apartados siguientes se ha obtenido de ahí, puesto que esa información es imprescindible para realizar y entender el diseño del módulo LCD_CTRL.

1.1 Secuencia para posicionar el cursor en un punto concreto

Para posicionar el cursor en un punto determinado, hay que enviar cierta información por la salida LCD_DATA: algunos comandos y datos de los mismos. Para que ese envío sea correcto, es necesario controlar también el valor del resto de las salidas.
Para enviar algo, sea comando o parámetro, se necesitan al menos 80 ns, es decir, 4 ciclos de reloj en nuestro sistema (f_{CLK}= 50 MHz; T_{CLK}= 20 ns). En el primer ciclo hay que activar las señales LCD_CS_N y LCD_WR_N (en lógica negativa), y desactivarlas en los tres ciclos siguientes. Además de eso, si lo que se envía en un comando, la señal LCD_RS tiene que tomar el valor L durante los 4 ciclos, pero si es un parámetro del comando, tiene que valer H.
De este modo, para posicionar el cursor en el punto indicado por las entradas XCOL e YROW hay que realizar los siguientes envíos (véase la Figura 2):

- Enviar el comando $2A_H$ (RS=L).
- Enviar el primer parámetro del comando 2A (RS=H): este parámetro siempre es 0.
- Enviar el segundo parámetro del comando 2A (RS=H): los 8 bits de mayor peso serán 0 y en los 8 bits de menos peso se enviará el valor de XCOL.
- Enviar el comando $2B_H$ (RS=L).
- Enviar el primer parámetro del comando 2B (RS=H): los 15 bits de más peso tienen valor 0 y el bit de menos peso toma el valor de $YROW_8$.
- Enviar el segundo parámetro del comando 2B (RS=H): los 8 bits de mayor peso serán 0 y en los 8 bits de menos peso serán $YROW_{7-0}$.

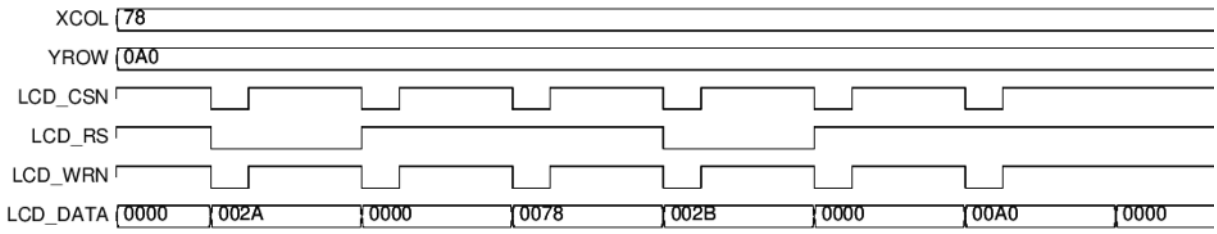

Figura 2

1.2 Dibujar pixels

Para colorear pixels de un determinado color hay que enviar cierta información por la salida LCD_DATA: un comando y tras él el valor del color tantas veces como pixels quieran colorearse. Igual que en el caso del cursor, un envío necesita 4 ciclos de reloj; en el primer ciclo se activan las señales LCD_CS_N y LCD_WR_N, y se desactivan en los tres siguientes; si lo que se envía es un comando LCD_RS=L y en otro caso H.
Por tanto, esta es la secuencia concreta que ha de enviarse por las líneas LCD_DATA, para dibujar NUM_PIX pixels del color RGB (véase la Figura 3):

- Comando $2C_H$ (RS=L).
- Parámetro del comando 2C: este parámetro indica el color del pixel y tendrá el valor de la entrada RGB. Cada vez que se envía se colorea un pixel, de modo que

para colorear NUM_PIX pixels habrá que enviar ese parámetro ese mismo número de veces. Como ejemplo, en la Figura 3 se envía 3 veces, lo que indica que se colorearán 3 pixels consecutivos.

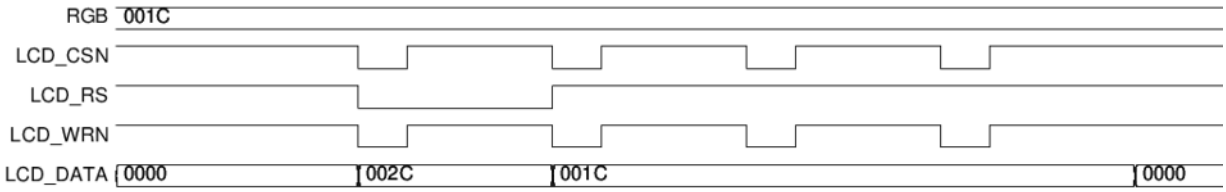

Figura 3

2. Diseño

De acuerdo con la metodología que se ha propuesto, diseñaremos el módulo mediante una unidad de proceso y unidad de control.

2.1 Unidad de proceso

El esquema de la Unidad de Proceso se representa en la Figura 4, y los componentes utilizados así como su función en el sistema son los siguientes:

- **3 registros** para guardar la información de entrada: XCOL, YROW y RGB.
- **Un contador descendente** para guardar la entrada NUMPIX. Lo decrementaremos con cada envío del color RGB para saber cuándo se ha finalizado el proceso de dibujar NUMPIX pixels.
- **Dos multiplexores** y **un contador** para conseguir el valor adecuado en cada momento en la salida LCD_DATA . El multiplexor de dos entradas lo utilizamos para poner en la salida o bien el valor 0 o bien el valor que viene de otro multiplexor de 8 entradas. En ese segundo multiplexor tenemos todas las opciones de envío, ya que en las entradas 0-5 se dispone de forma ordenada de la información que hay que enviar para posicionar el cursor (los dos comandos y los parámetros de cada uno); en las entradas 6-7, la información ordenada que hay que enviar para dibujar pixels (comando y parámetro). El orden es importante, ya que nos permite hacer la selección en el multiplexor a través de un contador: iremos incrementando el contador para ir cambiando la información en el momento adecuado.
- **Un descodificador** para analizar el valor del contador, ya que según sea el valor se tomarán varias decisiones en el algoritmo de control: continuar o no con los envíos, activar o no la señal RS etc.
- **Un registro** de 1 bit para determinar el valor de RS.

PROCESS UNIT

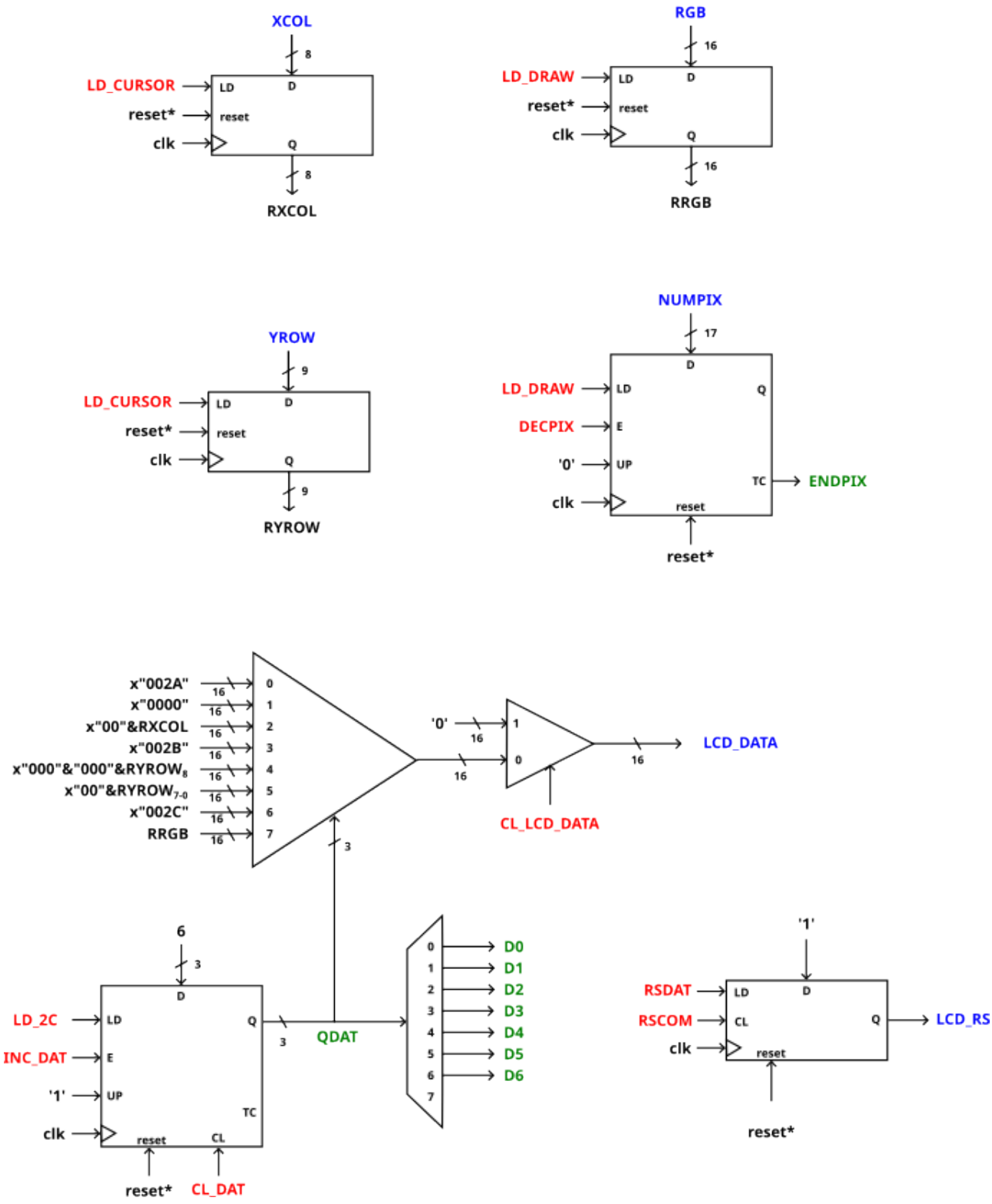

Figura 4

2.2 Algoritmo de control

El esquema del algoritmo de control está recogido en la figura 5 y las funciones principales de dicho algoritmo son las siguientes:

- En el estado E0 se asegura que la salida LCD_DATA está a 0 (CL_LCD_DATA activado) y se comprueba si la pantalla LCD ya ha sido inicializada (LT24_Init_Done). Si todavía no está inicializada no se analiza nada más. Si por el contrario ya está inicializada, se analiza si alguna de las operaciones básicas está siendo solicitada: señales OP_SETCURSOR y OP_DRAWCOLOUR. Si la petición de posicionamiento del curso está activada se continúa en el estado E1, y si la señal de dibujar pixels está activada se continúa en el estado E14.

- **Operación de posicionamiento del cursor.** Para posicionar el cursor hay que controlar el valor de las salidas LCD_WR_N, LCD_CS_N, LCD_RS y LCD_DATA. Las dos primeras las va a generar directamente la unidad de control, pero las otras vendrán del registro de 1 bit y de los multiplexores (ver esquema). La descripción de lo que hay que ir enviando por la salida LCD_DATA para posicionar el cursor está bien detallada en el apartado 1.1 y se corresponde con las 6 primeras entradas del multiplexor.
  Por lo tanto, después de realizar las inicializaciones necesarias en el estado E1, utilizaremos los estados E2, E3, E4, E11, E12 y E13 para realizar esos envíos mediante un bucle. Los tres primeros estados son iguales para todos los envíos, pero dependiendo del dato enviado, es decir, del analizando el valor de QDAT, el siguiente estado podrá ser E13, E12 o E11.
  El estado E11 es el último estado del bucle. A él se va tras haber realizado toda la secuencia de envíos , y por ello activamos en él la señal DONE_CURSOR.
  La diferencia entre E13 y E12 está en el control de la señal LCD_RS, ya que no tiene el mismo valor en todos los envíos (al enviar 2A o 2B su valor es L, pero al enviar los datos de esos comandos, debe valer H).
- **Operación de dibujar pixels**. Igual que para el cursor, para dibujar pixels hay que controlar el valor de las salidas LCD_WR_N, LCD_CS_N, LCD_RS y LCD_DATA. Sin embargo, en este caso la secuencia a enviar es más simple (véase el apartado 1.2). Una vez realizadas las las inicializaciones necesarias en el estado E14, se enviará el comando 2C mediante los estados E2, E3, E4 y E5, y después, dentro de un bucle, se utilizarán los estados E6, E7, E8 y E9 para dibujar el número de pixels solicitado. Para controlar el bucle se utiliza el contador descendente, que se va decrementando en cada iteración (DEC_PIX). Al llegar a 0 (ENDPIX) el bucle finaliza y se va al estado E10 en el que se activa la señal DONE_COLOUR.

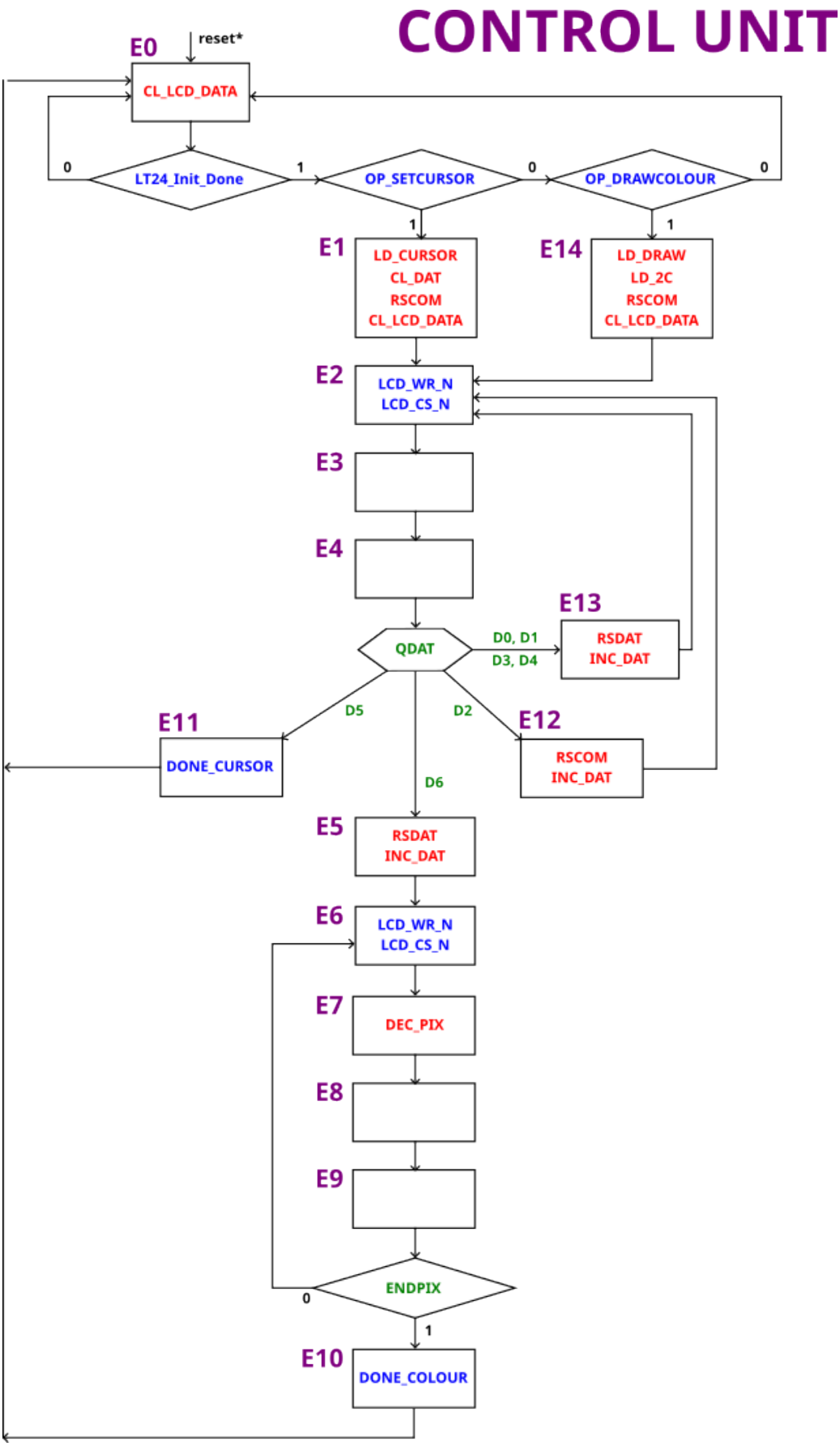
CONTROL UNIT
E0
reset*
CL_LCD_DATA
0
LT24_Init_Done
1
OP_SETCURSOR
0
OP_DRAWCOLOUR
0
1
1
E1
LD_CURSOR
CL_DAT
RSCOM
CL_LCD_DATA
E14
LD_DRAW
LD_2C
RSCOM
CL_LCD_DATA
E2
LCD_WR_N
LCD_CS_N
E3
E4
E13
QDAT
D0, D1
D3, D4
RSDAT
INC_DAT
D5
D2
E11
DONE_CURSOR
E12
RSCOM
INC_DAT
D6
E5
RSDAT
INC_DAT
E6
LCD_WR_N
LCD_CS_N
E7
DEC_PIX
E8
E9
ENDPIX
0
1
E10
DONE_COLOUR

Figura 5