

# LCD 16x2 en el NIOS II

Por Kalun José Lau Gan

1

## Changelog

- 26-06-2021 02:14 – Se separó del archivo de puertos debido al incremento de contenido.

2

## Índice:

- Aspectos iniciales del LCD alfanumérico HD44780
- El módulo Optrex 16207 LCD Controller Core del NIOS II
- Ejemplo de interface del NIOS II al LCD
- Creación de librería para el LCD

3

## El LCD alfanumérico

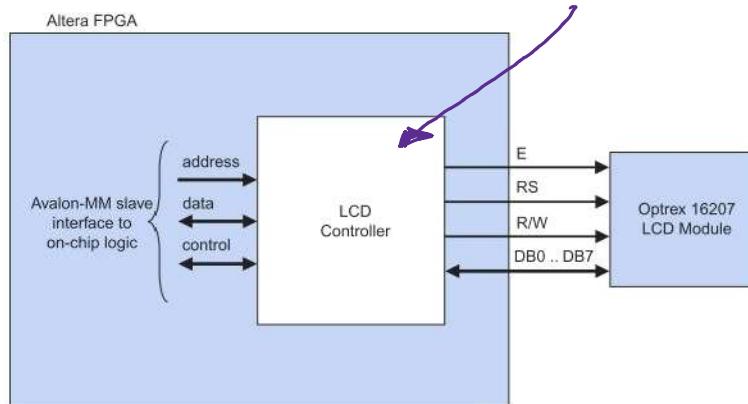


- Manejado por un controlador Hitachi HD44780
- Se pueden visualizar mensajes tanto en letras como en números.
- Diversos tamaños desde 1x8 hasta 4x40
- Su ROM de caracteres se asemeja a la tabla ASCII de 7bits
- Posee hasta ocho caracteres personalizados (CGRAM)
- Interface de 8 bits de datos + 3 líneas de control, opción para usar solo 4 bits de datos
- Hoja técnica:  
<https://www.sparkfun.com/datasheets/LCD/HD44780.pdf>

|          | 0000 | 0001 | 0010 | 0011    | 0100 | 0101    | 0110 | 0111 | 1000 | 1001 | 1010 | 1011 | 1100 | 1101 | 1110 | 1111 |
|----------|------|------|------|---------|------|---------|------|------|------|------|------|------|------|------|------|------|
|          | CS   | RS   | R/W  |         |      |         |      |      |      |      |      |      |      |      |      |      |
| xxxx0000 |      |      |      | 0@P`P   |      | -@E@P   |      |      |      |      |      |      |      |      |      |      |
| xxxx0001 | (2)  |      |      | !1AQaq  |      | ¤P@4äq  |      |      |      |      |      |      |      |      |      |      |
| xxxx0010 | (3)  |      |      | "2BRbr  |      | 「I@X@p@ |      |      |      |      |      |      |      |      |      |      |
| xxxx0011 | (4)  |      |      | #3CScs  |      | 』ウテ@e@  |      |      |      |      |      |      |      |      |      |      |
| xxxx0100 | (5)  |      |      | \$4DTdt |      | 、エト@μ@  |      |      |      |      |      |      |      |      |      |      |
| xxxx0101 | (6)  |      |      | %5EUeu  |      | ・オナユ@Ü  |      |      |      |      |      |      |      |      |      |      |
| xxxx0110 | (7)  |      |      | @6FUVfv |      | ヲカニヨ@Σ  |      |      |      |      |      |      |      |      |      |      |
| xxxx0111 | (8)  |      |      | *?GW@w  |      | アキヌラ@π  |      |      |      |      |      |      |      |      |      |      |
| xxxx1000 | (1)  |      |      | (8)H@hx |      | イタヌリ@x  |      |      |      |      |      |      |      |      |      |      |
| xxxx1001 | (2)  |      |      | >9IVi@  |      | カツル@y   |      |      |      |      |      |      |      |      |      |      |
| xxxx1010 | (3)  |      |      | *:JZjz  |      | エコ@レj#  |      |      |      |      |      |      |      |      |      |      |
| xxxx1011 | (4)  |      |      | +@K@k   |      | オサヒロ@n  |      |      |      |      |      |      |      |      |      |      |
| xxxx1100 | (5)  |      |      | ,<L@I@  |      | ヤシフワ@m  |      |      |      |      |      |      |      |      |      |      |
| xxxx1101 | (6)  |      |      | =M@M@   |      | ユスヘン@t  |      |      |      |      |      |      |      |      |      |      |
| xxxx1110 | (7)  |      |      | .@N@n@  |      | ヨセキ@n   |      |      |      |      |      |      |      |      |      |      |
| xxxx1111 | (8)  |      |      | /?0_o@  |      | ソウマ@ö   |      |      |      |      |      |      |      |      |      |      |

4

## Módulo LCD del NIOSII: Optrex 16207 LCD Controller Core



- Provee la interfaz en el NIOSII para comunicarse con un LCD alfanumérico basado en el controlador Hitachi HD44780.

5

## Módulo LCD del NIOSII: Optrex 16207 LCD Controller Core

- Revisar documentación de periféricos embebidos para NIOS II (ug\_embedded\_ip.pdf)

UG-01085 | 2020.09.21 [Send Feedback](#)

### 26. Optrex 16207 LCD Controller Core

#### 26.1. Core Overview

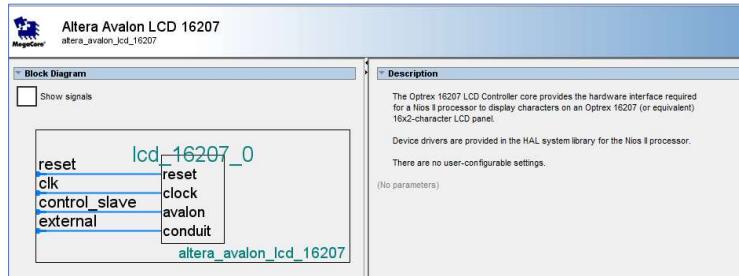
The Optrex 16207 LCD controller core with Avalon Interface (LCD controller core) provides the hardware interface and software driver required for a Nios II processor to display characters on an Optrex 16207 (or equivalent) 16x2-character LCD panel. Device drivers are provided in the HAL system library for the Nios II processor. Nios II programs access the LCD controller as a character mode device using ANSI C standard library routines, such as `printf()`. The LCD controller is Platform Designer-ready, and integrates easily into any Platform Designer-generated system.

The Nios II Embedded Design Suite (EDS) includes an Optrex LCD module and provides several ready-made example designs that display text on the Optrex 16207 via the LCD controller.

For details about the Optrex 16207 LCD module, see the manufacturer's *Dot Matrix Character LCD Module User's Manual* available online.

6

## Módulo LCD del NIOSII: Optrex 16207 LCD Controller Core



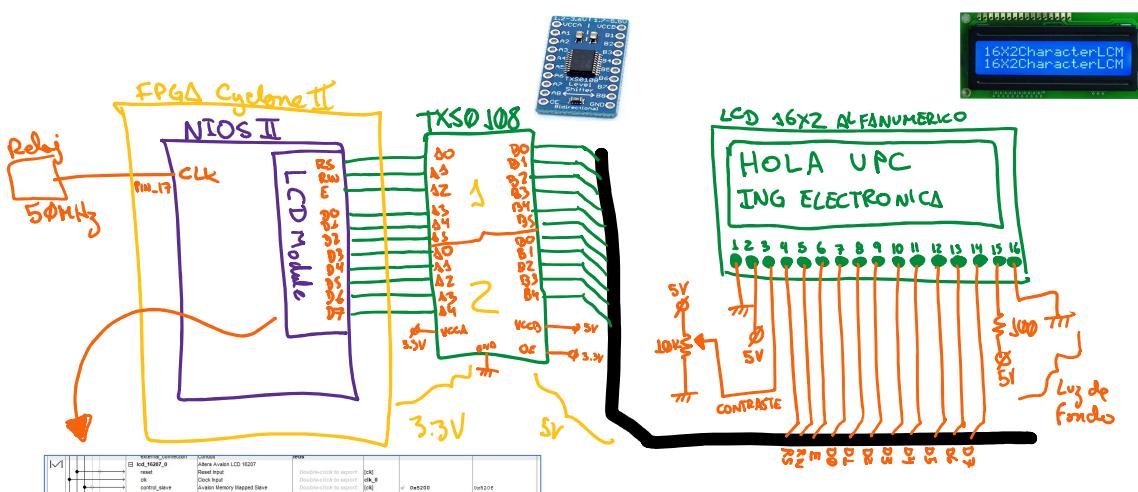
- Se instancia en el QSys, tener en consideración de no crear conflicto con los demás módulos instanciados.
  - Establecer nombre a la conexión externa para que se pueda conectar hacia pines físicos del FPGA en el PinPlanner



7

## Interface a display LCD alfanumérico con el FPGA Cyclone II EP2C5

- Tener en cuenta que los niveles lógicos de voltaje del FPGA son de 3.3V (LVTTL) y el LCD trabaja con niveles lógicos de voltaje de 5V (TTL).
- Se empleará módulos de conversión de niveles lógicos como el TXS0108 de ocho canales.



## Conexiones del módulo “Optrex 16207 LCD Controller Core” en el Quartus II

- Una vez generado la plataforma NIOS II en el Qsys incluyendo el módulo “Optrex 16207 LCD Controller Core” al instanciar el NIOS II como componente en el Quartus II aparecerán las señales a conectarse con el top-level:

```
component nios_upc is
  port(
    clk_clk      : in  std_logic          := 'X';           -- clk
    leds_export  : out std_logic_vector(7 downto 0);        -- export
    lcd1602_RS   : out std_logic;                      -- RS
    lcd1602_RW   : out std_logic;                      -- RW
    lcd1602_data : inout std_logic_vector(7 downto 0) := (others => 'X'); -- data
    lcd1602_E    : out std_logic;                      -- E
    botones_export : in  std_logic_vector(7 downto 0) := (others => 'X') -- export
  );
end component nios_upc;
```

9

## Conexiones del módulo “Optrex 16207 LCD Controller Core” en el Quartus II

- En la entidad donde se llama al componente NIOS II se deben de declarar las señales que conectarán los pines del FPGA con las señales del módulo “Optrex 16207 LCD Controller Core”

```
entity sem11_niosii_d is
  port(
    reloj:      in std_logic;
    foquitos:  out std_logic_vector(7 downto 0);
    display_rs: out std_logic;
    display_rw: out std_logic;
    display_e:  out std_logic;
    display_data: inout std_logic_vector(7 downto 0);
    buzzer:    out std_logic;
    disp7s_en:  out std_logic_vector(7 downto 0);
    botones:   in  std_logic_vector(7 downto 0)
  );
end sem11_niosii_d;
```

10

## Conexiones del módulo “Optrex 16207 LCD Controller Core” en el Quartus II

- En la declaración de conexiones (PORT MAP) se hacen las conexiones entre las señales del componente NIOS II con las señales de la entidad top-level

```

begin
    u0 : component nios_upc
        port map (
            clk_clk      => reloj,      -- clk.clk
            leds_export  => intermedia,  -- leds.export
            lcd1602_RS   => display_rs,  -- lcd1602.RS
            lcd1602_RW   => display_rw,  --      .RW
            lcd1602_data => display_data, --      .data
            lcd1602_E    => display_e,   --      .E
            botones_export => intermedia2 -- botones.export
        );
    
```

11

## Conexiones del módulo “Optrex 16207 LCD Controller Core” en el Quartus II

- A continuación todo el **código ejemplo** de la descripción estructural de un top-level para instanciar el procesador

```

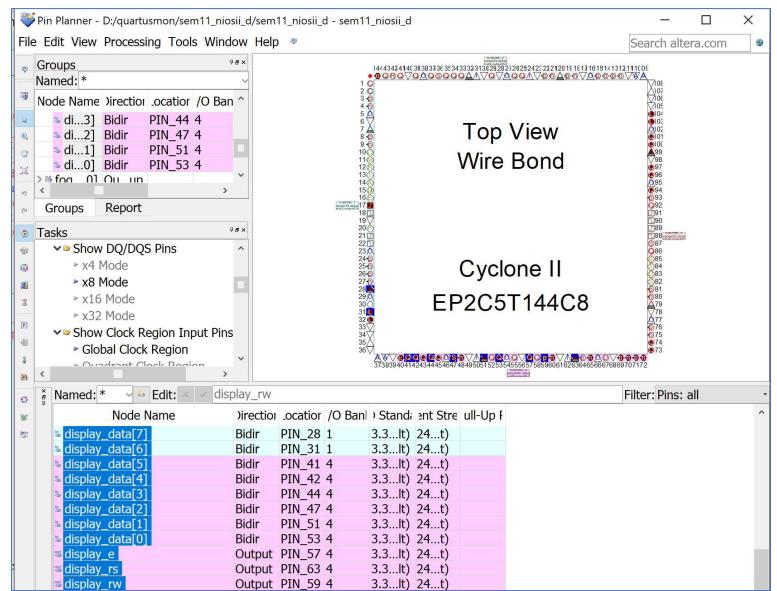
1 library IEEE;
2 use IEEE.std_logic_1164.all;
3
4 entity sem1_niosii_d is
5     port( reloj:  in std_logic;
6           foquitos: out std_logic_vector(7 downto 0);
7           display_rs: out std_logic;
8           display_rw: out std_logic;
9           display_e: out std_logic;
10          display_data: inout std_logic_vector(7 downto 0);
11          buzzer: out std_logic;
12          disp7s_en: out std_logic_vector(7 downto 0);
13          botones: in std_logic_vector(7 downto 0)
14      );
15 end sem1_niosii_d;
16
17 architecture flujo of sem1_niosii_d is
18
19 signal intermedia: std_logic_vector(7 downto 0);
20 signal intermedia2: std_logic_vector(7 downto 0);
21
22 component nios_upc is
23     port (
24         clk_clk      : in  std_logic          := 'X';
25         leds_export  : out std_logic_vector(7 downto 0);
26         lcd1602_RS   : out std_logic;
27         lcd1602_RW   : out std_logic;
28         lcd1602_data : inout std_logic_vector(7 downto 0) := (others => 'X');
29         lcd1602_E    : out std_logic;
30         botones_export : in  std_logic_vector(7 downto 0) := (others => 'X')
31     );
32 end component nios_upc;
33
34 begin
35     u0 : component nios_upc
36         port map (
37             clk_clk      -> reloj,      -- clk.clk
38             leds_export  => intermedia,  -- leds.export
39             lcd1602_RS   => display_rs,  -- lcd1602.RS
40             lcd1602_RW   => display_rw,  --      .RW
41             lcd1602_data => display_data, --      .data
42             lcd1602_E    => display_e,   --      .E
43             botones_export -> intermedia2 -- botones.export
44         );
45         buzzer <= '1';                      --Apagamos el molesto buzzer
46         foquitos <= not intermedia;
47         intermedia2 <= not botones;
48         disp7s_en <= (others => '1');       --Apagamos los displays de siete segmentos
49     end u0;
50 end flujo;

```

12

## Conexiones del módulo “Optrex 16207 LCD Controller Core” en el Quartus II

- Luego de una primera compilación del proyecto en el Quartus (luego de hacer el VHDL estructural instanciando el procesador y demás componentes) aparecerán los pines actualizados en el Pin Planner
- Tener especial atención a los pines que se van a conectar al LCD puesto a que **no todos los pines** disponibles en el FPGA pueden ser empleados para I/O.



13

## Grabación de la configuración en el FPGA

- Luego de la asignación de pines en el Pin Planner se procederá a realizar una segunda compilación del proyecto para que incluya las asignaciones de pines.
- Se procederá a grabar la configuración (archivo SOF) al FPGA mediante el programador del Quartus II.
- Luego de una grabación satisfactoria se seguirá el desarrollo en software Eclipse.

14

## Inicialización del LCD en el C para NIOS II:

- El LCD estará funcionando en modo 8 bits de datos (total 11 líneas que se conectan al NIOS II)
- Esta función se ejecuta por única al inicio de operación de la aplicación.
- Tener en cuenta que al inicio del código se debe incluir la librería y definiciones siguientes:

```
#include "altera_avalon_lcd_16207_regs.h"
```

```
#define LCD_WR_COMMAND_REG 0
#define LCD_RD_STATUS_REG 1
#define LCD_WR_DATA_REG 2
#define LCD_RD_DATA_REG 3
#define LCD_0_BASE 0x5200
```

dirección asignada  
en Qsys

```
void lcd_init(void) {
    usleep(15000); /* Wait for more than 15 ms before init */

    /* Set function code four times - 8-bit, 2 line, 5x7 mode */
    IOWR(LCD_0_BASE, LCD_WR_COMMAND_REG, 0x38);
    usleep(4100); /* Wait for more than 4.1 ms */
    IOWR(LCD_0_BASE, LCD_WR_COMMAND_REG, 0x38);
    usleep(100); /* Wait for more than 100 us */
    IOWR(LCD_0_BASE, LCD_WR_COMMAND_REG, 0x38);
    usleep(5000); /* Wait for more than 100 us */
    IOWR(LCD_0_BASE, LCD_WR_COMMAND_REG, 0x38);
    usleep(100); /* Wait for more than 100 us */

    /* Set Display to OFF*/
    IOWR(LCD_0_BASE, LCD_WR_COMMAND_REG, 0x08);
    usleep(100);

    /* Set Display to ON */
    IOWR(LCD_0_BASE, LCD_WR_COMMAND_REG, 0x0C);
    usleep(100);

    /* Set Entry Mode - Cursor increment, display doesn't shift */
    IOWR(LCD_0_BASE, LCD_WR_COMMAND_REG, 0x06);
    usleep(100);

    /* Set the Cursor to the home position */
    IOWR(LCD_0_BASE, LCD_WR_COMMAND_REG, 0x02);
    usleep(2000);
    /* Display clear */
    IOWR(LCD_0_BASE, LCD_WR_COMMAND_REG, 0x01);
    usleep(2000);
}
```

15

## Comandos en el LCD alfanumérico

- 0x01 – Limpia el display
- 0x02 – Mueve el cursor al inicio de la primera línea
- 0x0C – Apaga el cursor
- 0x0E – Cursor visible estático
- 0x0F – Cursor visible parpadeando
- 0x10 – Mueve cursor una posición a la izquierda
- 0x14 – Mueve cursor una posición a la derecha
- 0xC0 – Mueve cursor al inicio de la segunda línea
- 0x94 – Mueve el cursor al inicio de la tercera línea
- 0xD4 – Mueve el cursor al inicio de la cuarta línea

} Para LCDs de  
cuatro líneas

16

## Visualización de mensajes en el display:

- Visualizar “hola” en la primera linea

```
void main(void){
    //Función para inicializar el LCD
    lcd_init();

    //Comando para ubicarnos al inicio de la primera línea
    IOWR(LCD_0_BASE, LCD_WR_COMMAND_REG, 0x02);
    usleep(2000);

    //Visualización del mensaje "hola"
    IOWR(LCD_0_BASE, LCD_WR_DATA_REG, 'h');
    usleep(100);
    IOWR(LCD_0_BASE, LCD_WR_DATA_REG, 'o');
    usleep(100);
    IOWR(LCD_0_BASE, LCD_WR_DATA_REG, 'l');
    usleep(100);
    IOWR(LCD_0_BASE, LCD_WR_DATA_REG, 'a');
    usleep(100);

    while(1);

    return(0);
}
```

17

## Visualización de mensajes en el display:

- Visualizar “hola” (cadena) en la primera línea

```
const unsigned char cadena[] = {"hola"};
int x_var = 0;

void main(void){
    //Función para inicializar el LCD
    lcd_init();

    //Comando para ubicarnos al inicio de la primera línea
    IOWR(LCD_0_BASE, LCD_WR_COMMAND_REG, 0x02);
    usleep(2000);

    //Visualización del mensaje "hola"
    for(x_var=0;v_var<4;x_var++){
        IOWR(LCD_0_BASE, LCD_WR_DATA_REG,cadena[x_var]);
        usleep(100);
    }
    while(1);
    return(0);
}
```

18

## Visualización de mensajes en el display:

- Parametrizando en una función la visualización de una cadena

```
void ESCRIBE_MENSAJE(const char *cadena,unsigned char tam)
{
    unsigned char i = 0;
    for(i = 0; i<tam; i++)
    {
        IOWR(LCD_0_BASE, LCD_WR_DATA_REG, cadena[i]);
        usleep(100);
    }
}
```

tam: Tamaño de la cadena

- Ejemplo

```
void main(void){
    //Funcion para inicializar el LCD
    lcd_init();

    //Comando para ubicarnos al inicio de la primera linea
    IOWR(LCD_0_BASE, LCD_WR_COMMAND_REG, 0x02);
    usleep(2000);

    //Visualización del mensaje "hola"
    ENVIA_MENSAJE("hola", 4);
    while(1);
    return(0);
}
```

19

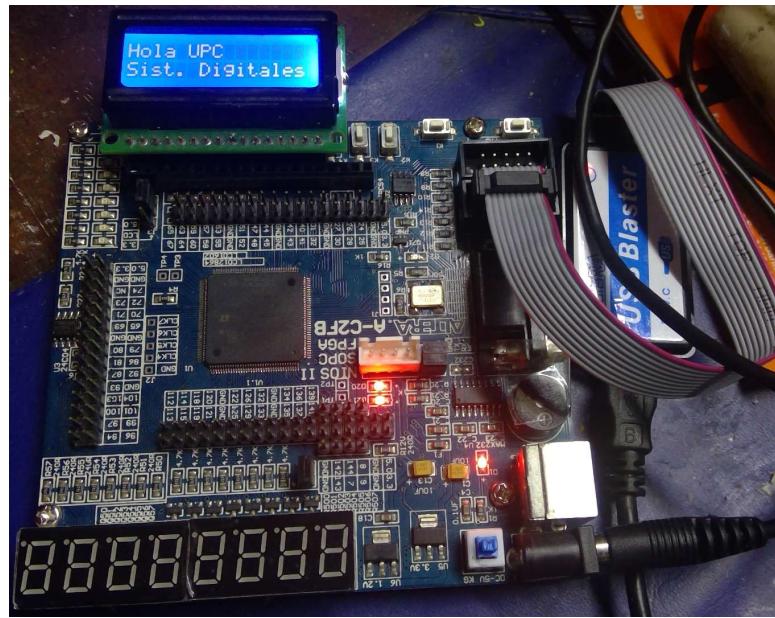
## Programa ejemplo completo:

```
1#include "sys/alt_stdio.h"
2#include "system.h"
3#include "altera_avalon_lcd_16207_regs.h"
4
5#define LCD_WR_COMMAND_REG 0
6#define LCD_RD_STATUS_REG 1
7#define LCD_WR_DATA_REG 2
8#define LCD_RD_DATA_REG 3
9#define LCD_0_BASE 0x5200
10
11void lcd_init(void){
12    usleep(15000);
13    IOWR(LCD_0_BASE, LCD_WR_COMMAND_REG, 0x38);
14    usleep(4100);
15    IOWR(LCD_0_BASE, LCD_WR_COMMAND_REG, 0x38);
16    usleep(100);
17    IOWR(LCD_0_BASE, LCD_WR_COMMAND_REG, 0x38);
18    usleep(5000);
19    IOWR(LCD_0_BASE, LCD_WR_COMMAND_REG, 0x38);
20    usleep(100);
21    IOWR(LCD_0_BASE, LCD_WR_COMMAND_REG, 0x08);
22    usleep(100);
23    IOWR(LCD_0_BASE, LCD_WR_COMMAND_REG, 0x0C);
24    usleep(100);
25    IOWR(LCD_0_BASE, LCD_WR_COMMAND_REG, 0x06);
26    usleep(100);
27    IOWR(LCD_0_BASE, LCD_WR_COMMAND_REG, 0x02);
28    usleep(2000);
29    IOWR(LCD_0_BASE, LCD_WR_COMMAND_REG, 0x01);
30    usleep(2000);
31}

33void ESCRIBE_MENSAJE(const char *cadena,unsigned char tam)
34{
35    unsigned char i = 0;
36    for(i = 0; i<tam; i++)
37    {
38        IOWR(LCD_0_BASE, LCD_WR_DATA_REG, cadena[i]);
39        | usleep(100);
40    }
41}
42
43void main(void)
44{
45    lcd_init();
46    IOWR(LCD_0_BASE, LCD_WR_COMMAND_REG, 0x02); //Primera linea
47    usleep(2000);
48    ESCRIBE_MENSAJE("Hola UPC", 8);
49    IOWR(LCD_0_BASE, LCD_WR_COMMAND_REG, 0xC0); //Segunda linea
50    usleep(2000);
51    ESCRIBE_MENSAJE("Sist. Digitales", 15);
52    while (1);
53}
```

20

## Implementación en la tarjeta A-C2FB



21

## Consideraciones adicionales

- Tener en cuenta que si se desea visualizar el contenido de alguna variable se tendrá que obtener los dígitos individuales (unidad, decena, centena, etc) para que pueda ser enviado uno por uno al display LCD.
- Un ejemplo de rutina de individualización de dígitos:

```

42 void convierte(unsigned int numero){
43     d_millar = numero / 1000;
44     millar = (numero % 1000) / 1000;
45     centena = (numero % 100) / 100;
46     decena = (numero % 10) / 10;
47     unidad = numero % 10;
48 }
```

22

## Creación de una librería para el manejo del LCD

- Al parametrizar las funciones relacionadas con el manejo del LCD las podemos agrupar en una librería.
- Con esto hacemos que el programa principal de la aplicación (el main.c) sea mucho mas compacto y fácil de entender.
- La librería creada podrá ser empleada en otros programas que requieran usar un LCD y por consiguiente aminorar el tiempo de desarrollo.
- En el lenguaje C se requiere que la librería tenga dos archivos:
  - La cabecera (\*.h) donde se especifican las cabeceras de las funciones.
  - El cuerpo (\*.c) donde se detalla el contenido de cada función.
- Se crearán entonces los archivos lcd\_lib.h y lcd\_lib.c a continuación.

23

## Creación de una librería para el manejo del LCD

- Archivo lcd\_lib.h:

```

8#ifndef LCD_LIB_H_
9#define LCD_LIB_H_
10
11#define LEFT 0
12#define RIGHT 1
13
14#include "sys/alt_stdio.h"
15#include "system.h"
16#include "unistd.h"
17#include "string.h"
18#include "altera_avalon_lcd_16207_regs.h"
19
20#define LCD_WR_COMMAND_REG 0
21#define LCD_RD_STATUS_REG 1
22#define LCD_WR_DATA_REG 2
23#define LCD_RD_DATA_REG 3
24#define LCD_0_BASE 0x4000
25
26void LCD_INIT(void);
27void LCD_ESCRIBE_MENSAJE(const char *cadena);
28void LCD_ENVIACHAR(unsigned char caracter);
29void LCD_CLEAR(void);
30void LCD_LINE1(void);
31void LCD_LINE2(void);
32void LCD_CUSTOM_CHAR(const unsigned char *vector, unsigned char pos);
33void LCD_CHARVAR_SEND(unsigned char numero, unsigned char show_digit);
34void LCD_POS_CURSOR(unsigned char fila,unsigned char columna);
35void LCD_CURSOR_MOVE(unsigned char direction);
36
37#endif /* LCD_LIB_H_ */

```

Dirección del LCD Core

Funciones para el manejo del LCD

24

# Creación de una librería para el manejo del LCD

- Archivo lcd\_lib.c:

```

8 #include "lcd.lib.h"
9
10 void LCD_INIT(void){
11     usleep(15000);
12     IOWR(LCD_0_BASE, LCD_WR_COMMAND_REG, 0x38);
13     usleep(4100);
14     IOWR(LCD_0_BASE, LCD_WR_COMMAND_REG, 0x38);
15     usleep(100);
16     IOWR(LCD_0_BASE, LCD_WR_COMMAND_REG, 0x38);
17     usleep(5000);
18     IOWR(LCD_0_BASE, LCD_WR_COMMAND_REG, 0x38);
19     usleep(100);
20     IOWR(LCD_0_BASE, LCD_WR_COMMAND_REG, 0x08);
21     usleep(100);
22     IOWR(LCD_0_BASE, LCD_WR_COMMAND_REG, 0x0C);
23     usleep(100);
24     IOWR(LCD_0_BASE, LCD_WR_COMMAND_REG, 0x06);
25     usleep(100);
26     IOWR(LCD_0_BASE, LCD_WR_COMMAND_REG, 0x02);
27     usleep(2000);
28     IOWR(LCD_0_BASE, LCD_WR_COMMAND_REG, 0x01);
29     usleep(2000);
30 }
31
32 void LCD_ESCRIBE_MENSAJE(const char *cadena) {
33     unsigned char tam;
34     tam = strlen(cadena);
35     unsigned char i=0;
36     for(i=0;i<tam;i++){
37         IOWR(LCD_0_BASE, LCD_WR_DATA_REG, cadena[i]);
38         usleep(100);
39     }
40 }
41
42 void LCD_ENVIACHAR(unsigned char caracter){
43     IOWR(LCD_0_BASE, LCD_WR_DATA_REG, caracter);
44     usleep(100);
45 }
46
47 void LCD_CLEAR(void) {
48     IOWR(LCD_0_BASE, LCD_WR_COMMAND_REG, 0x01);
49     usleep(2000);
50 }
51
52 void LCD_LINE1(void){
53     IOWR(LCD_0_BASE, LCD_WR_COMMAND_REG, 0x02);
54     usleep(2000);
55 }
56
57 void LCD_LINE2(void){
58     IOWR(LCD_0_BASE, LCD_WR_COMMAND_REG, 0xC0);
59     usleep(2000);
60 }
61
62 void LCD_CUSTOM_CHAR(const unsigned char *vector, unsigned char pos){
63     unsigned char i;
64     IOWR(LCD_0_BASE, LCD_WR_COMMAND_REG, 0x40+8*pos);
65     for(i=0;i<8;i++){
66         LCD_ENVIACHAR(vector[i]);
67     }
68     IOWR(LCD_0_BASE, LCD_WR_COMMAND_REG, 0x80);
69 }
70
71 void LCD_CHARVAR_SEND(unsigned char numero, unsigned char show_digit){
72     switch(show_digit){
73     case 1:
74         LCD_ENVIACHAR((numero%10)+0x30);
75         break;
76     case 2:
77         LCD_ENVIACHAR(((numero%10)/10)+0x30);
78         LCD_ENVIACHAR((numero%10)+0x30);
79         break;
80     case 3:
81         LCD_ENVIACHAR(((numero%1000)/100)+0x30);
82         LCD_ENVIACHAR(((numero%100)/10)+0x30);
83         LCD_ENVIACHAR((numero%10)+0x30);
84         break;
85     }
86 }
87
88 void LCD_POS_CURSOR(unsigned char fila,unsigned char columna){
89     if(fila == 1){
90         IOWR(LCD_0_BASE, LCD_WR_COMMAND_REG, 0x80+columna);
91     }
92     else if(fila == 2){
93         IOWR(LCD_0_BASE, LCD_WR_COMMAND_REG, 0xC0+columna);
94     }
95 }
96
97 void LCD_CURSOR_MOVE(unsigned char direction){
98     if (direction == LEFT){
99         IOWR(LCD_0_BASE, LCD_WR_COMMAND_REG, 0x10);
100    }
101    if (direction == RIGHT){
102        IOWR(LCD_0_BASE, LCD_WR_COMMAND_REG, 0x14);
103    }
104 }

```

25

# Creación de una librería para el manejo del LCD

- Detalle de cada función:

- LCD\_INIT(void)
  - Para la inicializar el LCD, se debe de llamar al inicio del programa de aplicación.
- LCD\_ESCRIBE\_MENSAJE(const char \*cadena)
  - Para visualizar una cadena de caracteres en pantalla.
- LCD\_ENVIACHAR(unsigned char caracter)
  - Para visualizar un solo carácter en la pantalla.
- LCD\_CLEAR(void)
  - Para limpiar la pantalla.
- LCD\_LINE1(void)
  - Para que el cursor se ubique al inicio de la primera línea.
- LCD\_LINE2(void)
  - Para que el cursor se ubique al inicio de la segunda línea.
- LCD\_CUSTOM\_CHAR(const unsigned char \*vector, unsigned char pos)
  - Para cargar un carácter personalizado (8 bytes) en una posición (pos) de la CGRAM destinada para albergar los caracteres personalizados (total 8 posiciones del 0 al 7).
- LCD\_CHARVAR\_SEND(unsigned char numero, unsigned char show\_digit)
  - Para imprimir una variable del tipo char en la pantalla con configuración de cuántos dígitos a imprimir.
- LCD\_POS\_CURSOR(unsigned char fila,unsigned char columna)
  - Para mover el cursor a determinada posición de determinara línea
- LCD\_CURSOR\_MOVE(unsigned char direction)
  - Para mover el curso a una posición a la izquierda o derecha dependiendo del parámetro de entrada (LEFT o RIGHT).

26

## Caracteres personalizados en el LCD 16x2



- El controlador HD44780 del LCD16x2 permite hasta ocho caracteres personalizados a la vez.
- Dichos 8 caracteres personalizados se alojan en las posiciones 0 al 7.

| Upper 4 bits | Lower 4 bits | Character | Description |
|--------------|--------------|-----------|-------------|
| xxxx0000     | 0001         | 0         | P           |
| xxxx0001     | 0010         | ! 1       | A           |
| xxxx0010     | 0011         | " 2       | B           |
| xxxx0011     | 0100         | # 3       | C           |
| xxxx0100     | 0101         | \$ 4      | D           |
| xxxx0101     | 0110         | % 5       | E           |
| xxxx0110     | 0111         | & 6       | F           |
| xxxx0111     | 1000         | ' 7       | G           |
| xxxx1000     | 1001         | ( 8       | H           |
| xxxx1001     | 1010         | ) 9       | I           |
| xxxx1010     | 1011         | * : J     | Z           |
| xxxx1011     | 1100         | + ; K     | [           |
| xxxx1100     | 1101         | , < L     | ]           |
| xxxx1101     | 1110         | - = M     | ]           |
| xxxx1110     | 1111         | . > N     | ^           |
| xxxx1111     | 0000         | / ? O     | _           |

27

## Caracteres personalizados en el LCD 16x2

- Para el diseño de un carácter personalizado se cuenta con un área de 8x5 pixeles, cada línea representa un dato. Si se desea activar un pixel simplemente se coloca un uno.
- Se necesitarán ocho datos para que pueda ingresarse un carácter personalizado a la memoria del LCD

| CGRAM Address                    | bit 4         | bit 3 | bit 2 | bit 1 | bit 0 | BIN         | HEX  |
|----------------------------------|---------------|-------|-------|-------|-------|-------------|------|
| 1 <sup>st</sup> Custom Character | byte 0 (0x40) |       |       |       |       | 0bxxx0 0100 | 0x04 |
|                                  | byte 1 (0x41) |       |       |       |       | 0bxxx0 1110 | 0xOE |
|                                  | byte 2 (0x42) |       |       |       |       | 0bxxx0 1110 | 0xOE |
|                                  | byte 3 (0x43) |       |       |       |       | 0bxxx0 1110 | 0xOE |
|                                  | byte 4 (0x44) |       |       |       |       | 0bxxx0 1110 | 0xOE |
|                                  | byte 5 (0x45) |       |       |       |       | 0bxxx1 1111 | 0x1F |
|                                  | byte 6 (0x46) |       |       |       |       | 0bxxx0 0100 | 0x04 |
|                                  | byte 7 (0x47) |       |       |       |       | 0bxxx0 0000 | 0x00 |

28

## Caracteres personalizados en el LCD 16x2

- Se recomienda usar el Excel para elaborar los caracteres personalizados.
- A continuación se muestra ocho caracteres personalizados de un reloj con manecillas en diferentes posiciones.

|    |  |      |    |  |      |     |  |      |     |  |      |
|----|--|------|----|--|------|-----|--|------|-----|--|------|
| 57 |  | 0x0E | 75 |  | 0x0E | 93  |  | 0x0E | 109 |  | 0x0E |
| 58 |  | 0x15 | 76 |  | 0x11 | 94  |  | 0x11 | 110 |  | 0x11 |
| 59 |  | 0x15 | 77 |  | 0x11 | 95  |  | 0x11 | 111 |  | 0x11 |
| 60 |  | 0x15 | 78 |  | 0x17 | 96  |  | 0x15 | 112 |  | 0x1D |
| 61 |  | 0x11 | 79 |  | 0x11 | 97  |  | 0x15 | 113 |  | 0x11 |
| 62 |  | 0x11 | 80 |  | 0x11 | 98  |  | 0x15 | 114 |  | 0x11 |
| 63 |  | 0x0E | 81 |  | 0x0E | 99  |  | 0x0E | 115 |  | 0x0E |
| 64 |  |      | 82 |  | 0x00 | 100 |  |      | 116 |  |      |
| 65 |  |      | 83 |  |      |     |  |      | 117 |  | 0x0E |
| 66 |  | 0x0E | 84 |  | 0x0E | 101 |  | 0x0E | 118 |  | 0x11 |
| 67 |  | 0x11 | 85 |  | 0x11 | 102 |  | 0x11 | 119 |  | 0x19 |
| 68 |  | 0x13 | 86 |  | 0x11 | 103 |  | 0x11 | 120 |  | 0x15 |
| 69 |  | 0x15 | 87 |  | 0x15 | 104 |  | 0x15 | 121 |  | 0x11 |
| 70 |  | 0x11 | 88 |  | 0x13 | 105 |  | 0x19 | 122 |  | 0x11 |
| 71 |  | 0x11 | 89 |  | 0x11 | 106 |  | 0x11 | 123 |  | 0x0E |
| 72 |  | 0x0E | 90 |  | 0x0E | 107 |  | 0x0E |     |  |      |
| 73 |  |      | 91 |  | 0x00 |     |  |      |     |  |      |

29

## Caracteres personalizados en el LCD 16x2

- Para almacenar un carácter personalizado en alguna posición de la CGRAM del LCD se ha creado la función:

```
void LCD_CUSTOM_CHAR(const unsigned char *vector, unsigned char pos)
{
    unsigned char i;
    IOWR(LCD_0_BASE, LCD_WR_COMMAND_REG, 0x40+8*pos); //Dirección de la CGRAM +
    for(i=0;i<8;i++)
    {
        LCD_ENVIACHAR(vector[i]);
    }
    IOWR(LCD_0_BASE, LCD_WR_COMMAND_REG, 0x80); //Dirección de la DDRAM
}
```

- Cuando se llame a la función anterior se necesitará como parámetros de entrada un arreglo de 8 bytes declarado como constante y la posición dentro de las ocho posiciones para caracteres personalizados:

```
const unsigned char reloj0[]={0x0E,0x15,0x15,0x15,0x11,0x11,0x0E,0x00};
```

LCD\_CUSTOM\_CHAR(reloj0, 0);

30

## Caracteres personalizados en el LCD 16x2

- Para la visualización del carácter personalizado (luego de haberlo almacenado con la función LCD\_CUSTOM\_CHAR) se empleará la función LCD\_ENVIACHAR indicando una de las ocho posiciones designadas para caracteres personalizados (del 0 al 7):

