

Manejo de puertos PIO y LCD en el NIOS II

Por: Kalun José Lau Gan

2021

1

Revisión (changelog) del documento:

- 13-noviembre-2020: Redacción inicial del documento
- 26-mayo-2021: Se completó el detalle faltante de la asignación de las señales del módulo “Optrex 16207 LCD Controller Core” en el Quartus II. Se añadió mas ejemplos de uso de los PIO.

2

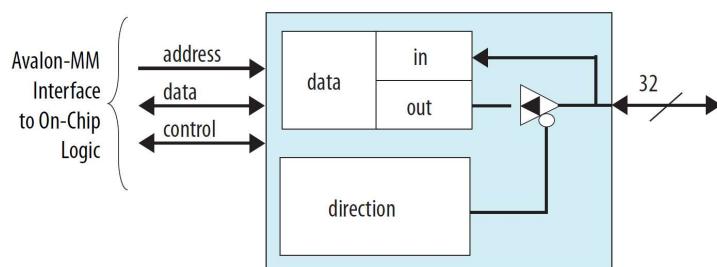
Agenda:

- Manipulación de puertos de entrada y salida (módulo PIO)
- Manejo de un LCD 16x2

3

Manipulación de puertos en el NIOS II

- Referencia: Manual de IPs de NIOS II, sección “PIO”.



- Revisar librería *altera_avalon_pio_regs.h*

4

Aspectos iniciales:

- Los PIO pueden configurarse de 1 bit a 32 bits de ancho.
- Pueden ser entradas, salidas o bidireccionales.
- Los registros implicados para el uso del PIO son de 32 bits mostrados a continuación:

Table 273. Register Map for the PIO Core

Offset	Register Name	R/W	(n-1)	...	2	1	0
0	data	read access	R	Data value currently on PIO inputs			
		write access	W	New value to drive on PIO outputs			
1	direction (1)	R/W		Individual direction control for each I/O port. A value of 0 sets the direction to input; 1 sets the direction to output.			
2	interruptmask (1)	R/W		IRQ enable/disable for each input port. Setting a bit to 1 enables interrupts for the corresponding port.			
3	edgecapture (1), (2)	R/W		Edge detection for each input port.			
4	outset	W		Specifies which bit of the output port to set. Outset value is not stored into a physical register in the IP core. Hence it's value is not reserve for future use.			
5	outclear	W		Specifies which output bit to clear. Outclear value is not stored into a physical register in the IP core. Hence it's value is not reserve for future use.			
Note :							
1. This register may not exist, depending on the hardware configuration. If a register is not present, reading the register returns an undefined value, and writing the register has no effect.							
2. If the option Enable bit-clearing for edge capture register is turned off, writing any value to the edgecapture register clears all bits in the register. Otherwise, writing a 1 to a particular bit in the register clears only that bit.							

5

Librería *altera_avalon_pio_regs.h*

```

31 #ifndef __ALTERA_AVALON_PIO_REGS_H__
32 #define __ALTERA_AVALON_PIO_REGS_H__
33
34 #include <iio.h>
35
36 #define IOADDR_ALTERA_AVALON_PIO_DATA(base)      __IO_CALC_ADDRESS_NATIVE(base, 0)
37 #define IORD_ALTERA_AVALON_PIO_DATA(base)        IORD(base, 0)
38 #define IOWR_ALTERA_AVALON_PIO_DATA(base, data)   IOWR(base, 0, data)
39
40 #define IOADDR_ALTERA_AVALON_PIO_DIRECTION(base)   __IO_CALC_ADDRESS_NATIVE(base, 1)
41 #define IORD_ALTERA_AVALON_PIO_DIRECTION(base)    IORD(base, 1)
42 #define IOWR_ALTERA_AVALON_PIO_DIRECTION(base, data) IOWR(base, 1, data)
43
44 #define IOADDR_ALTERA_AVALON_PIO_IRQ_MASK(base)   __IO_CALC_ADDRESS_NATIVE(base, 2)
45 #define IORD_ALTERA_AVALON_PIO_IRQ_MASK(base)    IORD(base, 2)
46 #define IOWR_ALTERA_AVALON_PIO_IRQ_MASK(base, data) IOWR(base, 2, data)
47
48 #define IOADDR_ALTERA_AVALON_PIO_EDGE_CAP(base)   __IO_CALC_ADDRESS_NATIVE(base, 3)
49 #define IORD_ALTERA_AVALON_PIO_EDGE_CAP(base)    IORD(base, 3)
50 #define IOWR_ALTERA_AVALON_PIO_EDGE_CAP(base, data) IOWR(base, 3, data)
51
52
53 #define IOADDR_ALTERA_AVALON_PIO_SET_BITS(base)   __IO_CALC_ADDRESS_NATIVE(base, 4)
54 #define IORD_ALTERA_AVALON_PIO_SET_BITS(base)    IORD(base, 4)
55 #define IOWR_ALTERA_AVALON_PIO_SET_BITS(base, data) IOWR(base, 4, data)
56
57 #define IOADDR_ALTERA_AVALON_PIO_CLEAR_BITS(base)  __IO_CALC_ADDRESS_NATIVE(base, 5)
58 #define IORD_ALTERA_AVALON_PIO_CLEAR_BITS(base)   IORD(base, 5)
59 #define IOWR_ALTERA_AVALON_PIO_CLEAR_BITS(base, data) IOWR(base, 5, data)
60
61
62
63 /* Definitions for direction-register operation with bi-directional PIOs */
64 #define ALTERA_AVALON_PIO_DIRECTION_INPUT 0
65 #define ALTERA_AVALON_PIO_DIRECTION_OUTPUT 1
66
67 #endif /* __ALTERA_AVALON_PIO_REGS_H__ */

```

6

Aspectos iniciales:

- Los PIO se declaran en el QSys
 - De acuerdo a los requerimientos se considerará puertos de entrada o de salida (no es recomendable trabajar con bidireccionales en esta etapa inicial).
 - Tener en cuenta la dirección asignada a cada PIO y revisar que no haya conflicto.
 - Asignar un nombre a la conexión externa del puerto para que aparezcan en el PinPlanner



7

Aspectos iniciales:

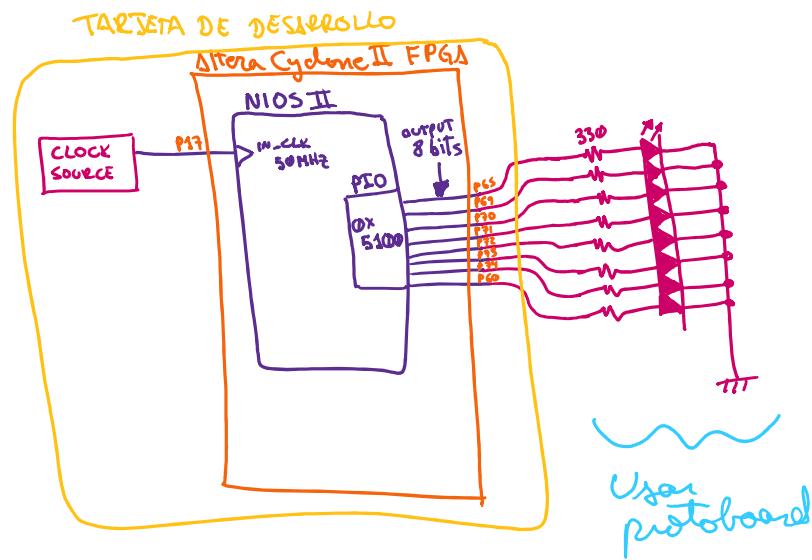
- Librería a incluir en el programa:
 - #include "altera_avalon_pio_regs.h"
- Escritura de dato en el PIO:
 - IOWR_ALTERA_AVALON_PIO_DATA([dirección del PIO], [dato]);
 - Ejemplo:


```
IOWR_ALTERA_AVALON_PIO_DATA(0x5100, 0x5A);
```
- Lectura de dato en el PIO:
 - IORD_ALTERA_AVALON_PIO_DATA([dirección del PIO]);
 - Ejemplo (se tendrá que declarar una variable donde se alojará lo leído):


```
data_in = IORD_ALTERA_AVALON_PIO_DATA(0x5200);
```

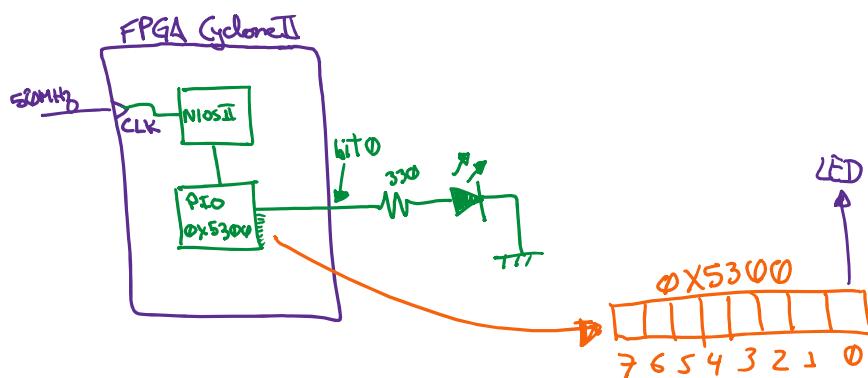
8

Ejemplo: PIO de salida de 8 bits en dirección 0x5100 conectado a ocho LEDs



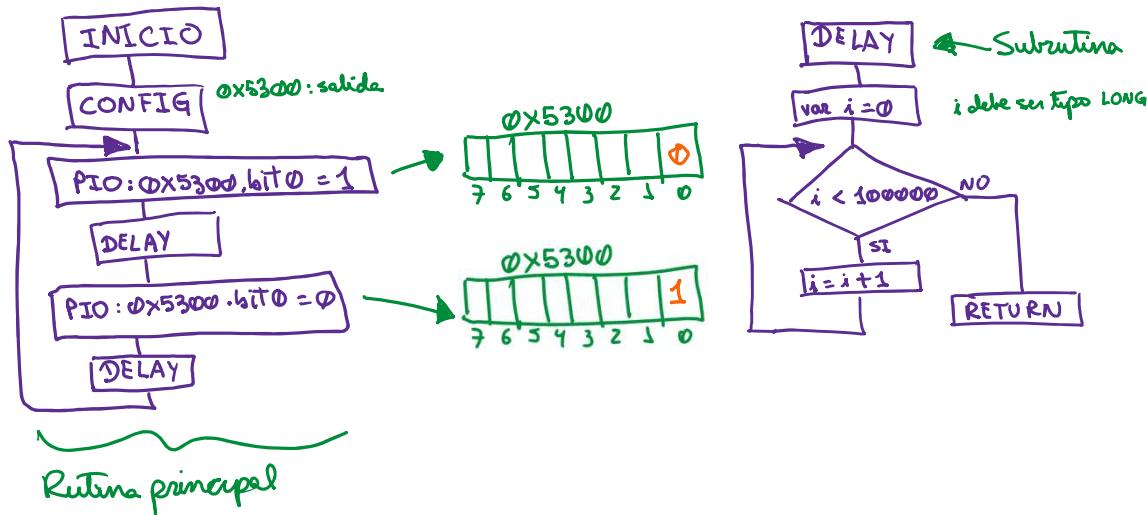
9

Ejemplo 1: Titilar un LED a través del bit0 del registro PIO de 8bits con dirección 0x5300



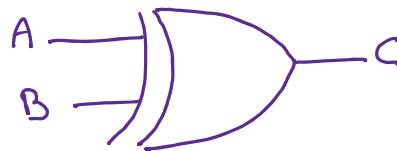
10

Ejemplo 1: Titilar un LED a través del bit0 del registro PIO de 8bits con dirección 0x5300

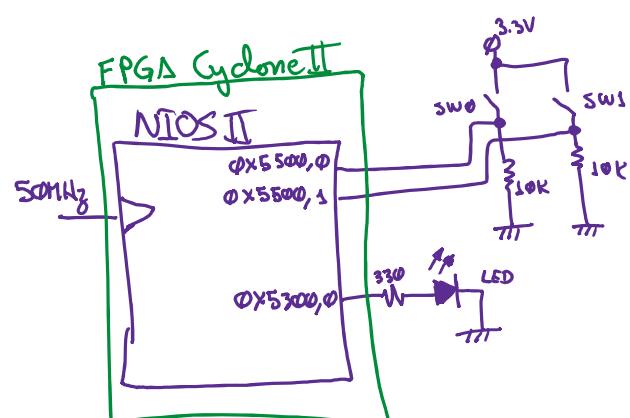


11

Ejemplo 2: Modelar una compuerta XOR en diagrama de flujo

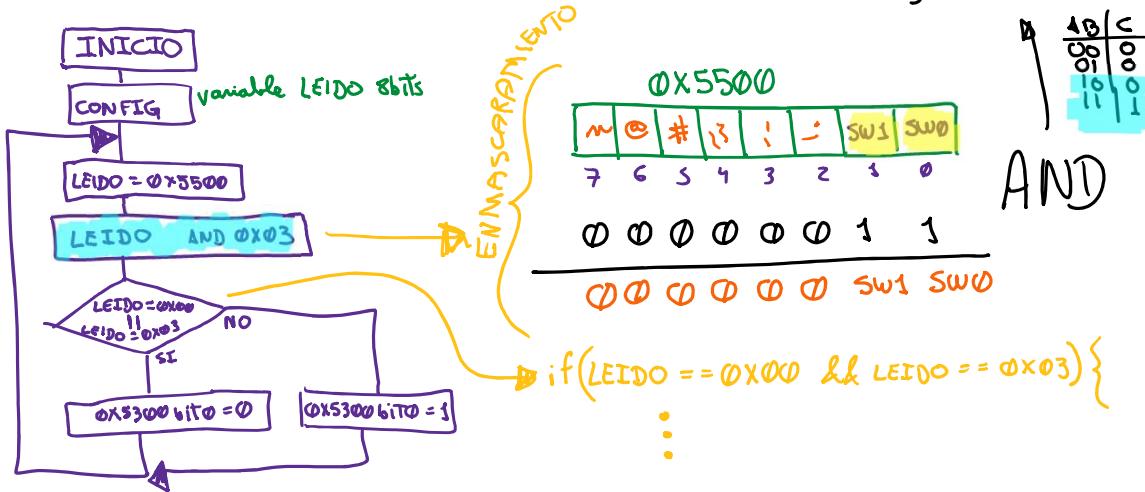


A	B	C
0	0	0
0	1	1
1	0	1
1	1	0



12

Ejemplo 2: Modelar una compuerta XOR en diagrama de flujo



13

Ejemplo 3: PIO de salida de 8 bits en dirección 0x5100

- El siguiente programa emite a través del PIO 0x5100 tres datos de manera secuencial y cíclica con un retardo definido por la subfunción `delay_s`
- Los datos son códigos en ASCII de la sigla UPC
- El comando `alt_putstr` se usa para enviar mensajes a la consola del NIOS II en el Eclipse

```

#include "sys/alt_stdio.h"
#include "system.h"
#include "altera_avalon_pio_regs.h"

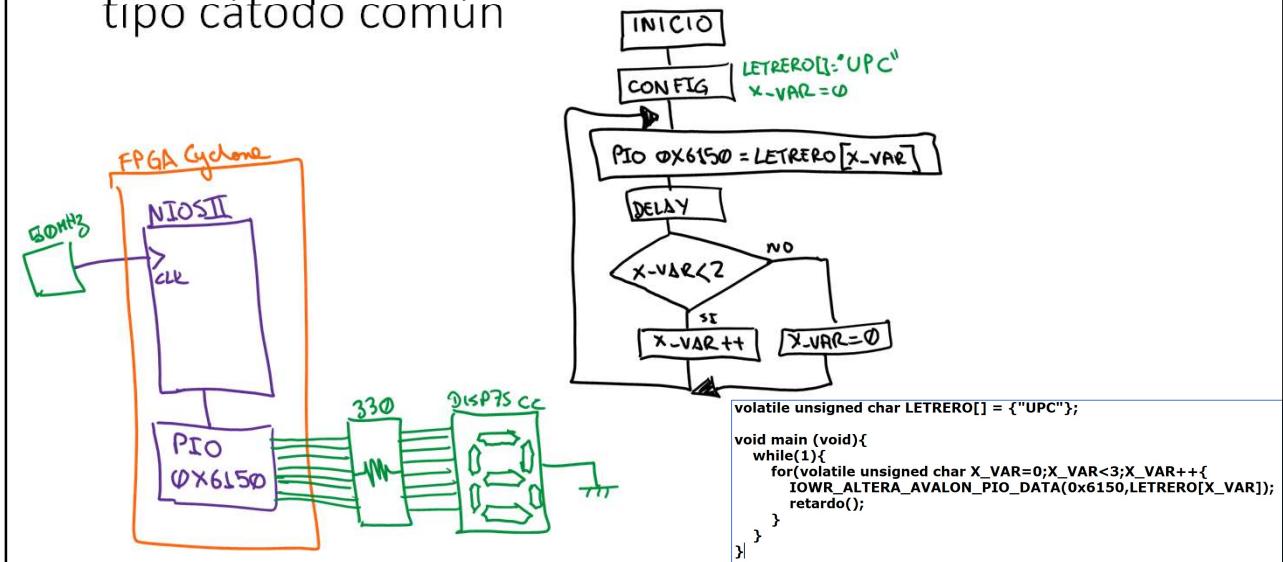
void delay_s(int tiempo){
    volatile int interno = 0;
    while (interno < tiempo*10000){
        interno++;
    }
}

int main()
{
    alt_putstr("Hello from Nios II!\n");
    /* Event loop never exits. */
    while (1){
        IOWR_ALTERA_AVALON_PIO_DATA(0x5100, 0x55); //Emite letra U
        delay_s(30);
        IOWR_ALTERA_AVALON_PIO_DATA(0x5100, 0x50); //Emite letra P
        delay_s(30);
        IOWR_ALTERA_AVALON_PIO_DATA(0x5100, 0x43); //Emite letra C
        delay_s(30);
    }
    return 0;
}

```

14

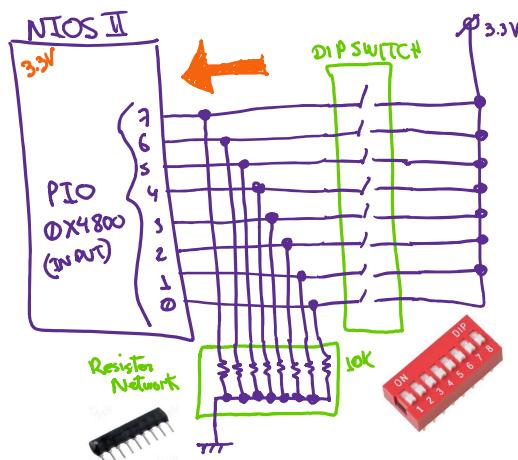
Ejemplo 4: Desarrollar un algoritmo para visualizar las siglas UPC en un display de siete segmentos del tipo cátodo común



15

Lectura de un PIO en el NIOS II

- Ejemplo5: Conectar un DIP switch en un PIO 0x4800



16

Lectura de un PIO en el NIOS II

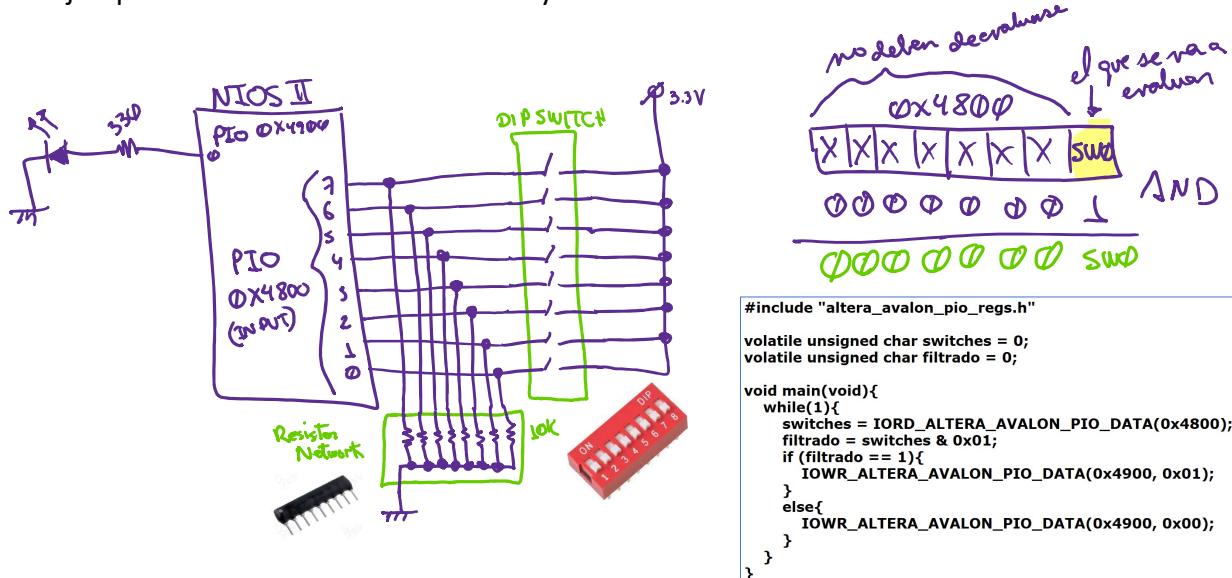
- Ejemplo5: Lectura del PIO en 0x4800 en código C del NIOSII

```
#include "altera_avalon_pio_regs.h"
volatile unsigned char switches = 0;
void main(void){
    while(1){
        switches = IORD_ALTERA_AVALON_PIO_DATA(0x4800);
    }
}
```

17

Lectura de un PIO en el NIOS II

- Ejemplo 5: Lectura de bit0 en 0x4800 y enviarlo a bit0 de 0x4900



18

Ejemplo 6: Desarrollar un negador lógico de un bit

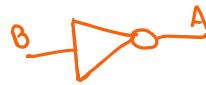
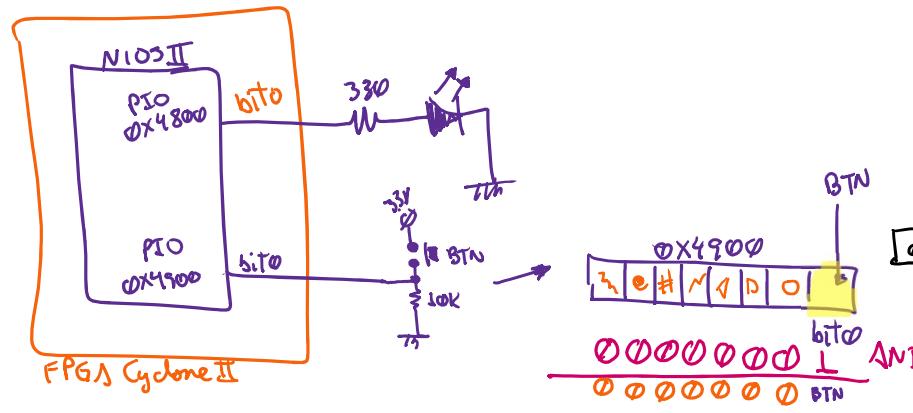
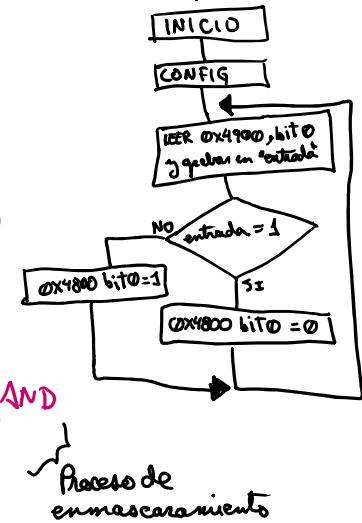


Diagrama de flujo:



19

Ejemplo 6: Desarrollar un negador lógico de un bit - Código en C para NIOS II:

```

#include "altera_avalon_pio_regs.h"

volatile unsigned char switches = 0;
volatile unsigned char filtrado = 0;

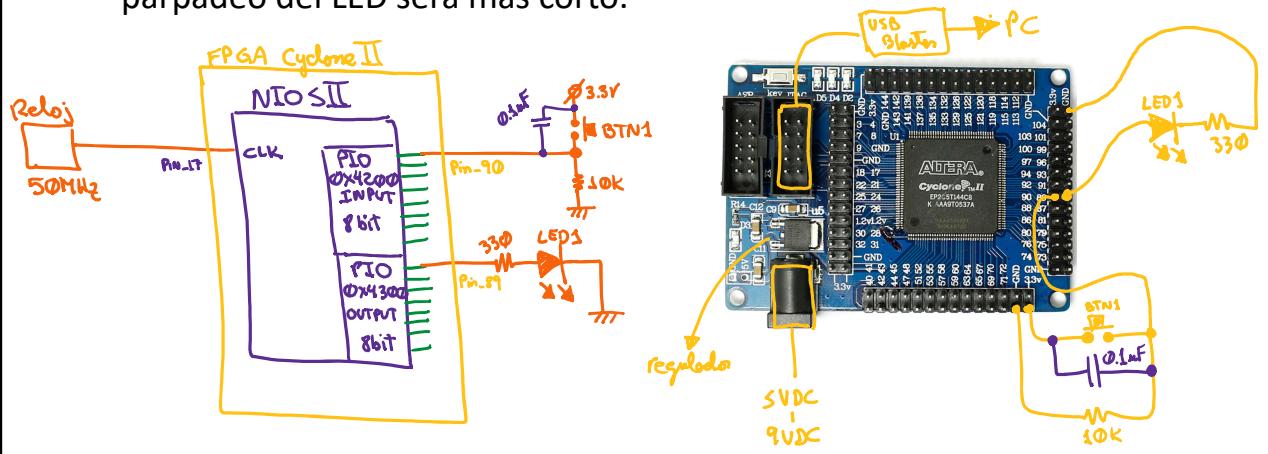
void main(void){
    while(1){
        switches = IORD_ALTERA_AVALON_PIO_DATA(0x4900);
        filtrado = switches & 0x01;
        if (filtrado == 1){
            IOWR_ALTERA_AVALON_PIO_DATA(0x4800, 0x00);
        }
        else{
            IOWR_ALTERA_AVALON_PIO_DATA(0x4800, 0x01);
        }
    }
}

```

20

Ejemplo 7: Lectura de PIO entradas

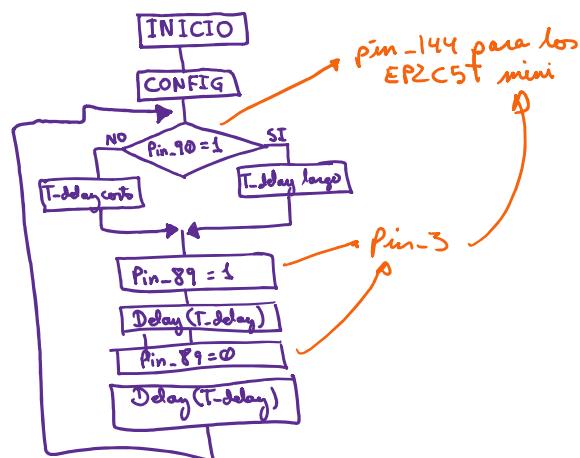
- En el siguiente circuito, cuando se presione el botón el tiempo de parpadeo del LED será mas corto.



21

Ejemplo 7: Lectura de PIO entradas

- Algoritmo en diagrama de flujo



22

Ejemplo 7: Lectura de PIO entradas

- Código en C para NIOS II

```
#include "sys/alt_stdio.h"
#include "altera_avalon_pio_regs.h"
#include "system.h"

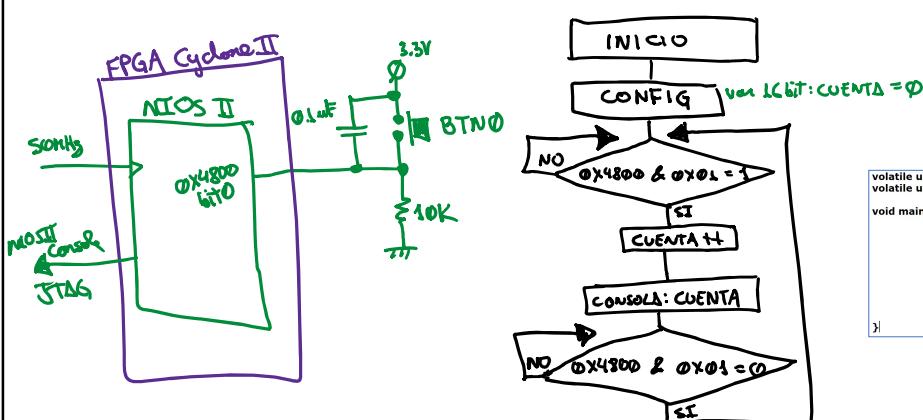
volatile unsigned char dato_in = 0;
volatile long tiempo = 0;

void main(void)
{
    alt_putstr("Hello from Nios II!\n");
    while(1){
        dato_in = (IORD_ALTERA_AVALON_PIO_DATA(0x4200)) & 0x01;
        if(dato_in == 1){
            tiempo = 200000;
        }
        else{
            tiempo = 100000;
        }
        IOWR_ALTERA_AVALON_PIO_DATA(0x4300, 0x01);
        usleep(tiempo);
        IOWR_ALTERA_AVALON_PIO_DATA(0x4300, 0x00);
        usleep(tiempo);
    }
    return 0;
}
```

Nota: Se está haciendo un enmascaramiento al momento de leer el PIO entrada para que solo se tome en cuenta el bit0 del puerto.

23

Ejemplo 8: Detectar la cantidad de veces que se presiona un botón y visualizarlo como mensaje en la consola del NIOS II



```

volatile unsigned char LEIDO = 0;
volatile unsigned int CUENTA = 0;

void main (void){
    while(1){
        LEIDO = IORD_ALTERA_AVALON_PIO_DATA(0x4800);
        LEIDO = LEIDO & 0x01
        if (LEIDO == 1){
            CUENTA = CUENTA + 1;
            alt_putstr(CUENTA);
        }
    }
    while(LEIDO == 1);
}
  
```

24

El LCD alfanumérico

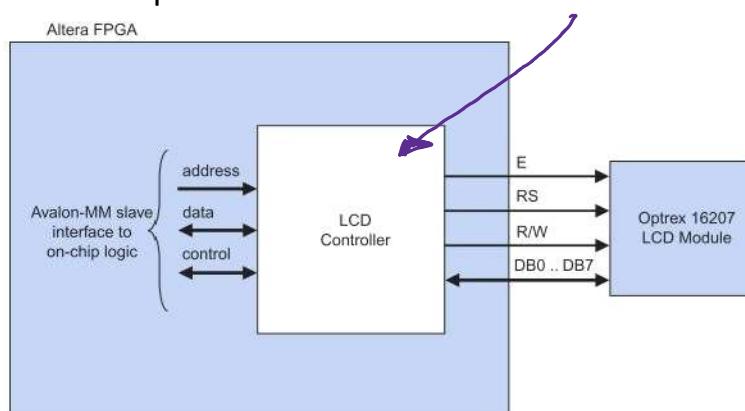


- Manejado por un controlador Hitachi HD44780
- Se pueden visualizar mensajes tanto en letras como en números.
- Diversos tamaños desde 1x8 hasta 4x40
- Su ROM de caracteres se asemeja a la tabla ASCII de 7bits
- Posee hasta ocho caracteres personalizados (CGRAM)
- Interface de 8 bits de datos + 3 líneas 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
xxxx0000	00P`P		-T3exp													
xxxx0001 (2)	.1AQaq		■748q													
xxxx0010 (3)	"2BRbr		■47zpe8													
xxxx0011 (4)	#3CScs		■9TE8e													
xxxx0100 (5)	\$4DTdt		■ITBmu													
xxxx0101 (6)	%5EUeu		■OANUgU													
xxxx0110 (7)	&6FUVfu		■KAN3pS													
xxxx0111 (8)	*?7GWgw		■KAN2gN													
xxxx1000 (1)	(8HXhx		■KAN1jX													
xxxx1001 (2))9IYi		■KAN1jY													
xxxx1010 (3)	*:JZjz		■KAN1jZ													
xxxx1011 (4)	+;KLk		■KAN1jK													
xxxx1100 (5)	,<L%I		■KAN1j%													
xxxx1101 (6)	-=M]m)		■KAN1jM													
xxxx1110 (7)	.>N^n+		■KAN1jN													
xxxx1111 (8)	/?0_o+		■KAN1jO													

25

Módulo LCD del NIOSII: Optrex 16207 LCD Controller Core



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

26

Módulo LCD del NIOSII: Optrex 16207 LCD Controller Core

- Revisar documentación de periféricos embebidos para NIOS II (ug_embedded_ip.pdf)

The screenshot shows a page from the Intel Nios II Embedded Design Suite documentation. At the top left is the document identifier "UG-01085 | 2020.09.21" and a "Send Feedback" button. At the top right is the Intel logo. Below the header, the title "26. Optrex 16207 LCD Controller Core" is underlined. A sub-section titled "26.1. Core Overview" follows. The main text describes the core's function: "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." It also notes that the Nios II EDS includes an Optrex LCD module and provides several ready-made example designs. A small note at the bottom states: "For details about the Optrex 16207 LCD module, see the manufacturer's *Dot Matrix Character LCD Module User's Manual* available online."

27

Módulo LCD del NIOSII: Optrex 16207 LCD Controller Core



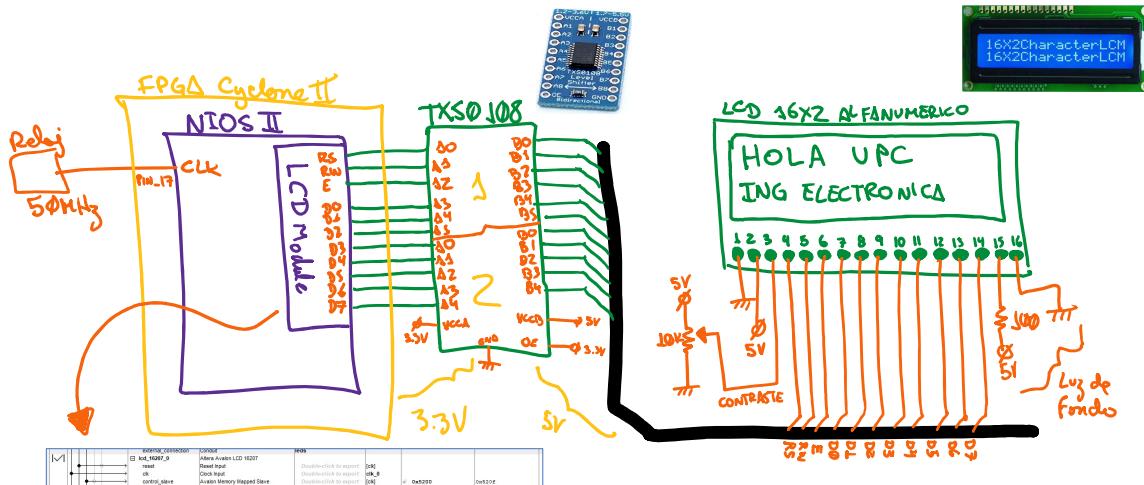
- Se instancia en el QSys, tener en consideración de no crear conflicto con los demás módulos instanciados.
 - Establecer nombre a la conexión externa para que se pueda conectar hacia pines físicos del FPGA en el PinPlanner



28

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.



29

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;
```

30

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;
```

31

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
    );

```

32

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

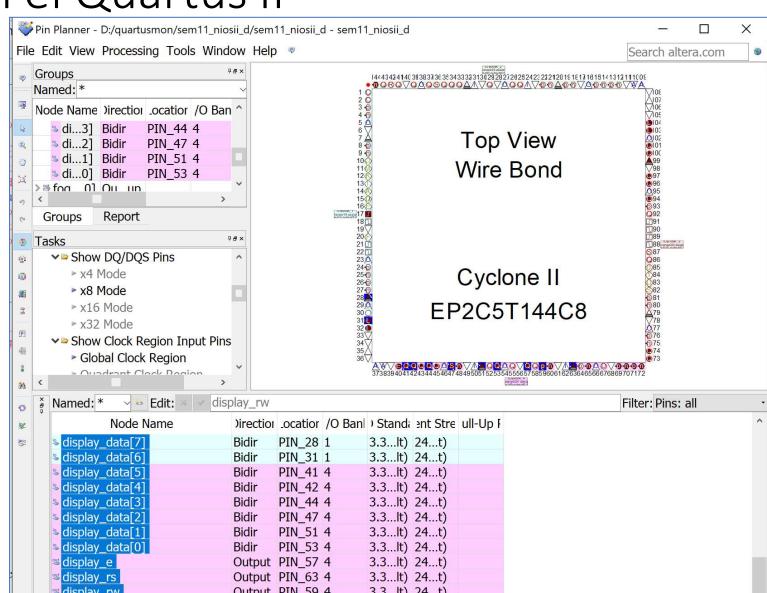
```

1 library IEEE;
2 use IEEE.std_logic_1164.all;
3
4 entity sem11_niosii_d is
5   port( reloj:  in std_logic;
6         fogueos: 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 sem11_niosii_d;
16
17 architecture flujo of sem11_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, -- lcd1602.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       fogueos <= not intermedia; --Apagamos los displays de siete segmentos
47       intermedia2 <= not botones;
48       disp7s_en <= (others => '1'); --Apagamos los displays de siete segmentos
49   end flujo;
50 
```

33

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



34

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.

35

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);
}
```

36

Comandos en el LCD alfanumérico

- 0x01 – Limpia el display
- 0x02 – Mueve el cursor al inicio de la primera línea
- 0x0C – Apaga el curso
- 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

37

Visualización de mensajes en el display:

- Visualizar “hola” en la primera linea

```
void main(void){
    //Funcion 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);
}
```

38

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){
    //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"
    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);
}
```

39

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);
}
```

40

Programa ejemplo completo:

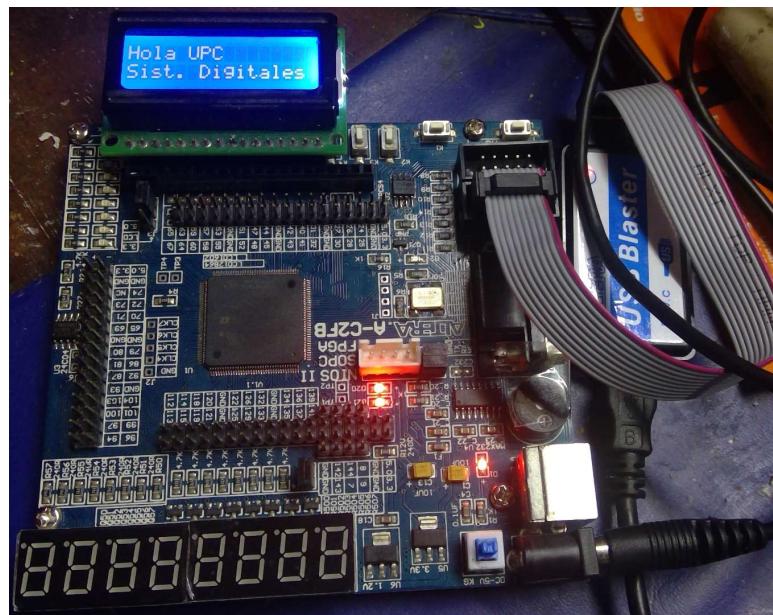
```

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
11 void 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 }
33 void 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
43 void 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 }

```

41

Implementación en la tarjeta A-C2FB



42

Consideraciones finales

- 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 / 10000;  
44     millar = (numero % 10000) / 1000;  
45     centena = (numero % 1000) / 100;  
46     decena = (numero % 100) / 10;  
47     unidad = numero % 10;  
48 }
```