Desarrollo de un Turnero Digital Usando FPGA's Laboratorio de Electrónica Digital Módulo: 3, RS232

Héctor F. Jiménez S.
hfjimenez@utp.edu.co
PGP KEY ID: 0xB05AD7B8

Sebastian Zapata
sebastzapata93@gmail.com
PGP KEY ID: 0xfffff

Francisco Gallego
juanfran16@utp.edu.co

Fecha de Entrega: **20** Octubre,2016
Profesor: Ing.Msc(c) Ramiro Andres Barrios Valencia

1 Objetivos

- Desarrollar un modulo de comunicación utilizando el protocolo Rs232, y vhdl.

2 Marco Teórico

En la actualidad existen una gran cantidad de formas de comunicación que hacen que sea posible el intercambio de datos entre 2 o más dispositivos electrónicos, para que esto suceda los dispositivos deben elegir un protocolo el cual permiten establecer los parámetros de la misma. El protocolo *Rs232* también conocido como EIA/TIA RS-232C designa una norma para el intercambio de datos binarios que se enviaran en forma serial, este tipo de comunicación en serie implica el envío de una serie de pulsos digitales de ida y vuelta entre dispositivos a una velocidad de transferencia mutuamente acordada.
Con el fin de hacer una comunicación serial, los dos dispositivos deben ponerse de acuerdo en los siguientes parámetros:

1. Velocidad a la que se envía, y leen los datos.

2. Los niveles de tensión que representan un 1 o un 0.

3. El significado de los niveles de tensión; Nivel de tensión alto es un **1** y un nivel de tensión bajo representa un **0**, o lógica inversa.

Figure 1: Interfaz Serial Usb-Db9.

Para la comunicación entre ambos, ademas de los parámetros mencionados anteriormente es necesario usar una interfaz serial, en el mercado hay varias soluciones que se adaptan dependiendo el tipo de puerto y entrada por la cual enviaremos, como se muestra en la figura 1 la solucion tambien se puede embeber en el mismo cable.
Una interfaz rs232 físicamente requiere :

- Una conexión a tierra común, ambos dispositivos tienen un punto de referencia.
- Un cable para el remitente para enviar datos al receptor (línea de transmisión para el remitente conocido como **TX**).
- Un cable para el remitente para recibir datos (línea de recepcion conocido como **RX**), véase la figura2. SparkFun[1].

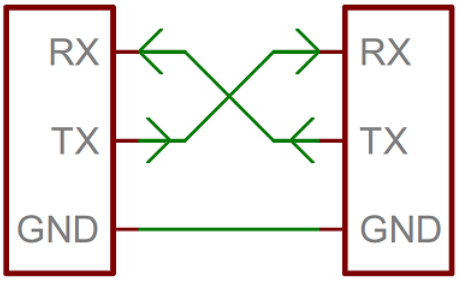

Figure 2: Hardware necesario para conexion asincrona.

Para realizar el envió de comunicación el protocolo rs232 sigue una maquina de estado que se resume en los siguientes pasos:

1. El transmisor envia un **"idle" (="1")** cuando esta en reposo.

1. `https://learn.sparkfun.com/tutorials/serial-communication`

2. El transmisor enviara un bit de inicio **"start" (="0")** que indica el comienzo de la transmision de datos.

3. Se envian los 8 bits de informacion

4. El transmisor envia un bit que indica la parada de cada byte.**"stop" (="1")** [2]

Figure 3: Hardware necesario para conexion asincrona.

Para desarrollar este modulo se crearon dos partes , la del receptor y el transmisor, cada uno se encarga de su tarea de enviar los datos serializado, recibirlos. El codigo implementado se presenta en las secciones de codigo 1, 2.

2. Vease la figura 3

```
process(clk,clk_enable,rx) begin
        if rising_edge(clk) then
                rx_temp<=rx;
                if clk_enable = '1' then
                        case rx_state is
                                when idle=>
                                        data_ready<='0';
                                        if rx_temp='0' then
                                                rx_data<=(others=>'1');
                                                rx_state<=data;
                                        end if;
                                when data=>
                                        data_ready<='0';
                                        rx_data<= (others=>'1');
                                        if rx_temp='1' then
                                                rx_parity_bit<=not(rx_parity_bit);
                                        end if;
                                        rx_out_temp(rx_data_counter)<=rx_temp;
                                        if rx_data_counter=7 then
                                                rx_data_counter<=0;
                                                rx_state<=parity_state;
                                        else
                                                rx_data_counter<=rx_data_counter+1;
                                        end if;
                                when parity_state=>
                                        data_ready<='0';
                                        rx_data<= (others=>'1');
                                        if rx_parity_bit=rx_temp then
                                                data_enable<='1';
                                        else
                                                data_enable<='0';
                                        end if;
                                        rx_state<=stop1;
                                when stop1=>
                                        data_ready<='0';
                                        rx_data<= (others=>'1');
                                        rx_state<=stop2;
                                when stop2=>
                                        if data_enable='1' then
                                                data_ready<='1';
                                                rx_data<=rx_out_temp;
                                        else
                                                data_ready<='0';
                                                rx_data<= (others=>'1');
                                        end if;
                                        rx_parity_bit<='1';
                                        rx_state<=idle;

                                end case;
                        end if;
        end if;
end process;
```

Listing 1: Modulo de Recepcion RX.

```
process(clk,clk_enable) begin
                if rising_edge(clk) then
                        if clk_enable = '1' then
                                tx_data_temp<=tx_data;
                                case tx_state is
                                        when idle=>
                                                if tx_start = '1' then
                                                        tx <= uart_start;
                                                        tx_state <= data;
                                                else
                                                        tx<=uart_idle;
                                                end if;
                                        when data=>
                                                tx <= tx_data_temp(tx_data_counter);
                                                if tx_data_temp(tx_data_counter)='1' then
                                                        tx_parity_bit<=not (tx_parity_bit);
                                                end if;
                                                if tx_data_counter = 7 then
                                                        tx_data_counter<=0;
                                                        tx_state<=parity_state;
                                                else
                                                        tx_data_counter<=tx_data_counter+1;
                                                end if;
                                        when parity_state=>
                                                tx <= tx_parity_bit;
                                                tx_state <=stop1;
                                        when stop1=>
                                                tx <=uart_idle;
                                                tx_state<=stop2;
                                        when stop2=>
                                                tx <=uart_idle;
                                                tx_parity_bit<='1';
                                                tx_state<=idle;
                                end case;
                        end if;
                end if;
        end process;
```

Listing 2: Modulo de Transmision TX uart.

3 Anexos

los demás archivos son adicionales que resuelven la misma implementación, se encuentran en el repositorio[3] *Nota*: Las referencias utilizadas se encuentran en los pies de página. Si requiere de manera detallada estas contacte con *miembros del equipo.*

3. `https://github.com/heticor915/DigitalElectronicsLab/tree/master/Reports/Module3`