Módulo 3: Protocolo RS232

Lady Johanna Trejos Hernández

Andrés Felipe Betancurt Rivera

Docente Ramiro Andrés Barrios Valencia

Universidad Tecnológica de Pereira

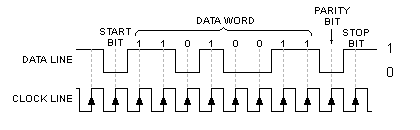
Facultad de Ingenierías FEECC

Asignatura Laboratorio de Electrónica Digital

20 de abril de 2017

RS-232 *(Recommended Standard 232, en español: "Estándar Recomendado 232")*, es una interfaz que designa una norma para el intercambio de datos binarios serie entre un DTE *(Data Terminal Equipment, "Equipo Terminal de Datos")* y un DCE *(Data Communication Equipment, "Equipo de Comunicación de Datos")*.

El formato de la transmisión de datos se indica en la figura:



Para iniciar una transmisión se debe enviar un cero en la señal indicando el bit de inicio *(start bit),* seguidamente se envían bit a bit los n bits del dato (5, 6, 7 u 8 bits), el tamaño es definido para todos los datos que recibirá el protocolo.

Al recibir el último bit del dato se envía un bit opcional denominado bit de paridad *(Parity bit)* usado para datos de 8 bits, para detectar errores en la transmisión.

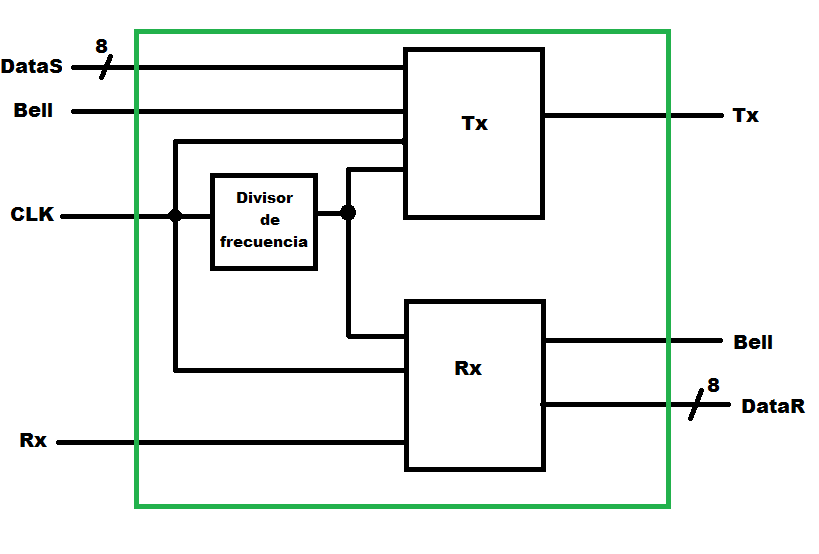
El bit de parada *(stop bit)* indica el final de la transmisión con la señal en 1. Se pueden usar uno o dos bits de parada.

Mientras el sistema no está recibiendo información, se transmite un 1 binario denominado idle.

Para el presente trabajo se utilizó el siguiente formato de transmisión:

* Dato de 8 bits, dado a que las lecturas sobre el protocolo concuerdan en transmitir y recibir elementos del código ASCII.
* Bit de paridad par, no se tendrá en cuenta el dato corrupto, no corregirá errores.
* Dos bits de parada.
* Frecuencia de 9600 baudios

El diagrama a seguir será el discutido en clase:



Se diseñaron de forma individual los tres módulos principales de este proyecto:

* **Divisor de frecuencia:**

Este divisor se creó a partir de la siguiente ecuación:

Para este caso:

Es decir, el periodo para el reloj debe ser 104160 ns.

***Código en VHDL***

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.STD\_LOGIC\_UNSIGNED.ALL;

entity Divisor is

Port ( clk : in STD\_LOGIC;

newC : out STD\_LOGIC);

end Divisor;

architecture Behavioral of Divisor is

--signal aux: integer range 0 to 433; --115200 (433, 217)

--signal aux: integer range 0 to 2603; --19200 (2603, 1302)

signal aux: integer range 0 to 5207; --9600 (5207, 2604)

begin

process (clk)

begin

if rising\_edge (clk) then

if (aux = 5207) then

aux <= 0;

elsif (aux < 2604 or aux > 2604) then

newC <= '0';

aux <= aux + 1;

elsif (aux = 2604) then

newC <= '1';

aux <= aux + 1;

end if;

end if;

end process;

end Behavioral;

* **Módulo Tx**

Para este módulo se sigue la explicación del docente, primero se definen las entradas:

* CLK: Reloj de la tarjeta de desarrollo
* Campana: Avisa que hay un dato para transmitir
* Dato\_entrada: Dato de 8 bits a transmitir (Sin protocolo)

Y la salida en serie:

* Dato\_salida: Transmisión del dato en forma serial con el protocolo implementado (Bit de inicio, n bits del dato, bit de paridad, bits de parada)

***Código en VHDL:***

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.STD\_LOGIC\_UNSIGNED.ALL;

entity Tx is

Port ( Campana : in STD\_LOGIC;

CLK : in STD\_LOGIC;

Dato\_entrada : in STD\_LOGIC\_VECTOR (7 downto 0);

Dato\_salida : out STD\_LOGIC);

end Tx;

architecture Behavioral of Tx is

COMPONENT Divisor

PORT(

clk : IN std\_logic;

newC : OUT std\_logic

);

END COMPONENT;

signal reloj\_baudios : std\_logic := '0';

signal estado : std\_logic\_vector(1 downto 0) := "00";

-- 00 -> Salida Idle

-- 01 -> Envío de dato

-- 10 -> Envío de bit de paridad

-- 11 -> Envío de bits de parada

signal contador\_de\_bits : std\_logic\_vector(3 downto 0) := "0000";

signal Paridad : std\_logic := '0';

signal aux : std\_logic := '1'; -- inicializar la salida en '1', en idle.

begin

Inst\_Divisor: Divisor PORT MAP(

clk => CLK,

newC => reloj\_baudios

);

Dato\_salida <= aux;

process (CLK)

variable Dato\_temporal : std\_logic\_vector(7 downto 0) := (others => '0');

begin

if (rising\_edge(CLK) and reloj\_baudios = '1') then

if (estado = "00") then

aux <= '1';

if (Campana = '1') then

Dato\_temporal := Dato\_entrada;

estado <= "01";

aux <= '0';

end if;

* **Módulo Rx**

elsif (estado = "01") then

if (Dato\_temporal(0) = '1') then

Paridad <= not Paridad;

end if;

aux <= Dato\_temporal(0);

Dato\_temporal := '0' & Dato\_temporal(7 downto 1);

contador\_de\_bits <= contador\_de\_bits + 1;

if (contador\_de\_bits >= "0111") then

contador\_de\_bits <= (others => '0');

estado <= "10";

end if;

elsif (estado = "10") then

aux <= Paridad;

estado <= "11";

elsif (estado = "11") then

if (contador\_de\_bits = "0001") then

estado <= "00";

contador\_de\_bits <= (others => '0');

aux <= '1';

Paridad <= '0';

else

aux <= '1';

contador\_de\_bits <= contador\_de\_bits + 1;

end if;

end if;

end if;

end process;

end Behavioral;

Entradas:

* CLK: Reloj de la tarjeta de desarrollo
* Dato\_entrada: Dato de 8 bits recibido de forma serial con el protocolo requerido

Y la salida en serie:

* Campana: Campana para avisar que llegó un dato
* Dato\_salida: Dato recibido sin protocolo

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.STD\_LOGIC\_UNSIGNED.ALL;

entity Rx is

Port ( Dato\_entrada : in STD\_LOGIC;

CLK : in STD\_LOGIC;

Dato\_salida : out STD\_LOGIC\_VECTOR (7 downto 0);

Campana : out STD\_LOGIC);

end Rx;

architecture Behavioral of Rx is

COMPONENT Divisor

PORT( clk : IN std\_logic;

newC : OUT std\_logic);

END COMPONENT;

signal reloj\_baudios : std\_logic := '0';

signal estado : std\_logic\_vector(1 downto 0) := "00";

-- 00 -> Recepción de Idle

-- 01 -> Recepción de dato

-- 10 -> bit de paridad

-- 11 -> bits de parada

signal contador\_de\_bits : std\_logic\_vector(3 downto 0) := "0000";

signal Paridad : std\_logic := '0';

signal aux : std\_logic\_vector(7 downto 0) := "00000000";

begin

Inst\_Divisor: Divisor PORT MAP(

clk => CLK,

newC => reloj\_baudios);

Dato\_salida <= aux;

process (CLK)

variable Dato\_temporal : std\_logic\_vector(7 downto 0) := (others => '0');

begin

if (rising\_edge(CLK) and reloj\_baudios = '1') then

Campana <= '0';

if (estado = "00" and Dato\_entrada = '0') then

estado <= "01";

elsif (estado = "01") then

if (Dato\_entrada = '1') then

Paridad <= not Paridad;

end if;

Para finalizar, se añaden los tres módulos a uno solo llamado “ProtocoloRS232.vhd”

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity ProtocoloRS232\_v2 is

Port ( Tx\_salida : out STD\_LOGIC;

Rx\_entrada : in STD\_LOGIC;

CLK : in STD\_LOGIC;

CampanaTx : in STD\_LOGIC;

CampanaRx : out STD\_LOGIC;

Dato\_Tx : in STD\_LOGIC\_VECTOR (7 downto 0);

Dato\_Rx : out STD\_LOGIC\_VECTOR (7 downto 0));

end ProtocoloRS232\_v2;

Dato\_temporal(6 downto 0) := Dato\_temporal(7 downto 1);

Dato\_temporal(7) := Dato\_entrada;

contador\_de\_bits <= contador\_de\_bits + 1;

if (contador\_de\_bits >= "0111") then

contador\_de\_bits <= (others => '0');

estado <= "10";

end if;

elsif (estado = "10") then

if (Paridad = Dato\_entrada) then

estado <= "11";

else

estado <= "00";

Paridad <= '0';

end if;

elsif (estado = "11") then

if (contador\_de\_bits = "0001") then

estado <= "00";

contador\_de\_bits <= (others => '0');

aux <= Dato\_temporal;

Campana <= '1';

elsif (Dato\_entrada = '1') then

contador\_de\_bits <= contador\_de\_bits + 1;

else

estado <= "00";

Paridad <= '0';

end if;

end if;

end if;

end process;

end Behavioral;

architecture Behavioral of ProtocoloRS232\_v2 is

COMPONENT Rx

PORT(

Dato\_entrada : IN std\_logic;

CLK : IN std\_logic;

Dato\_salida : OUT std\_logic\_vector(7 downto 0);

Campana : OUT std\_logic

);

END COMPONENT;

COMPONENT Tx

PORT(

Campana : IN std\_logic;

CLK : IN std\_logic;

Dato\_entrada : IN std\_logic\_vector(7 downto 0);

Dato\_salida : OUT std\_logic

);

END COMPONENT;

begin

Inst\_Rx: Rx PORT MAP(

Dato\_entrada => Rx\_entrada,

CLK => CLK,

Dato\_salida => Dato\_Rx,

Campana => CampanaRx

);

Inst\_Tx: Tx PORT MAP(

Campana => CampanaTx,

CLK => CLK,

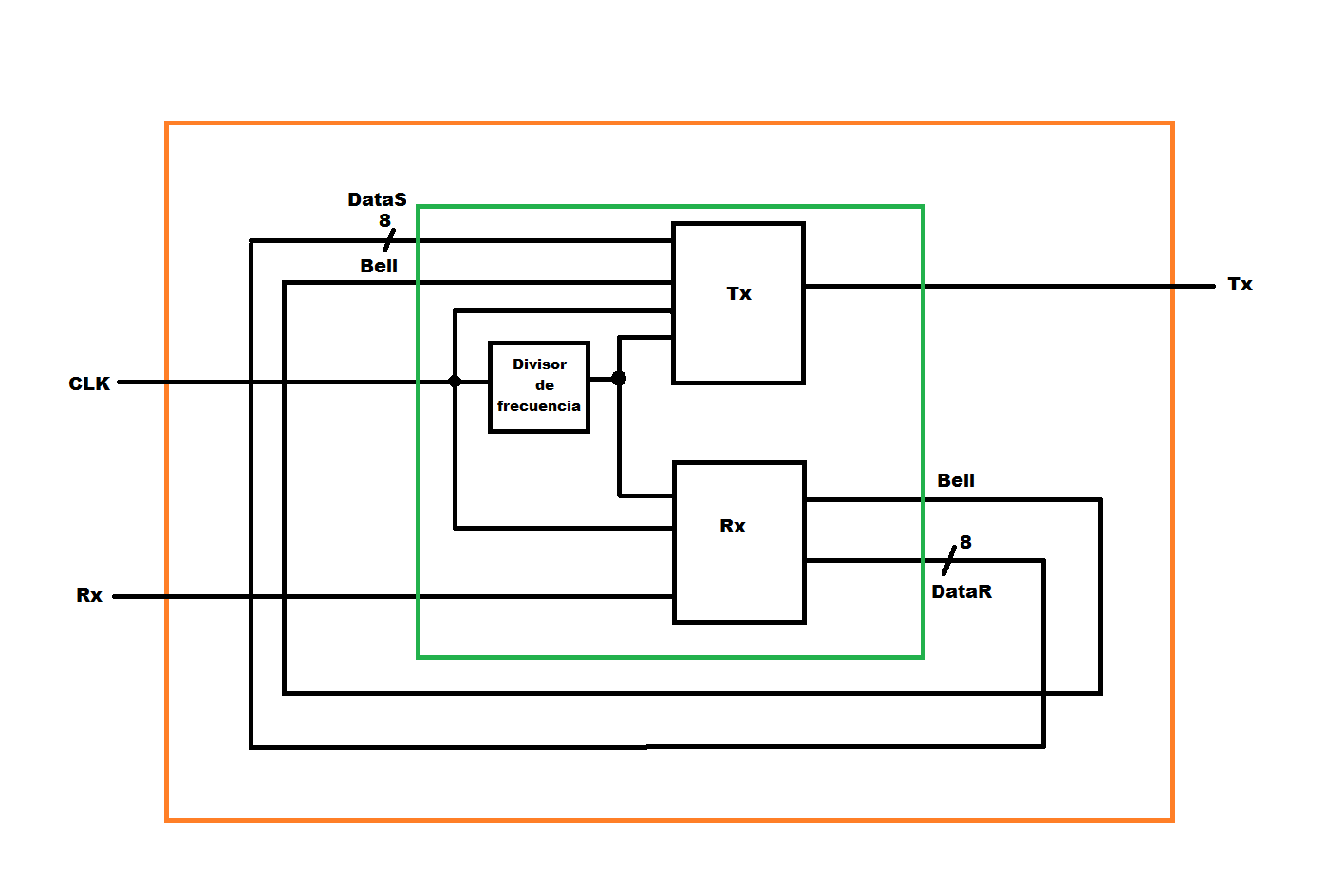
Dato\_entrada => Dato\_Tx,

Dato\_salida => Tx\_salida

);

end Behavioral;

Para la entrega del módulo 3, se conectan las campanas y los datos de 8 bits para simular un ECO como se indica en la imagen.



library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity ECO is

Port ( Rx : in STD\_LOGIC;

Tx : out STD\_LOGIC;

CLK : in STD\_LOGIC);

end ECO;

architecture Behavioral of ECO is

COMPONENT ProtocoloRS232\_v2

PORT(

Rx\_entrada : IN std\_logic;

CLK : IN std\_logic;

CampanaTx : IN std\_logic;

Dato\_Tx : IN std\_logic\_vector(7 downto 0);

Tx\_salida : OUT std\_logic;

CampanaRx : OUT std\_logic;

Dato\_Rx : OUT std\_logic\_vector(7 downto 0)

);

END COMPONENT;

signal campana : std\_logic := '0';

signal Dato : std\_logic\_vector(7 downto 0) := "00000000";

begin

Inst\_ProtocoloRS232\_v2: ProtocoloRS232\_v2 PORT MAP(

Tx\_salida => Tx,

Rx\_entrada => Rx,

CLK => CLK,

CampanaTx => campana,

CampanaRx => campana,

Dato\_Tx => Dato,

Dato\_Rx => Dato

);

end Behavioral;

**TEST BENCH**

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

ENTITY ECO\_TB IS

END ECO\_TB;

ARCHITECTURE behavior OF ECO\_TB IS

-- Component Declaration for the Unit Under Test (UUT)

COMPONENT ECO

PORT(

Rx : IN std\_logic;

Tx : OUT std\_logic;

CLK : IN std\_logic

);

END COMPONENT;

--Inputs

signal Rx : std\_logic := '0';

signal CLK : std\_logic := '0';

--Outputs

signal Tx : std\_logic;

-- Clock period definitions

constant CLK\_period : time := 20 ns;

BEGIN

-- Instantiate the Unit Under Test (UUT)

uut: ECO PORT MAP (

Rx => Rx,

Tx => Tx,

CLK => CLK

);

-- Clock process definitions

CLK\_process :process

begin

CLK <= '0';

wait for CLK\_period/2;

CLK <= '1';

wait for CLK\_period/2;

end process;

-- Stimulus process

stim\_proc: process

begin

Rx <= '1';

wait for 10 ns;

wait for 156240 ns;

Rx <= '0';

wait for 104160 ns;

Rx <= '1';

wait for 104160 ns;

Rx <= '0';

wait for 104160 ns;

Rx <= '1';

wait for 104160 ns;

Rx <= '0';

wait for 104160 ns;

Rx <= '1';

wait for 104160 ns;

Rx <= '1';

wait for 104160 ns;

Rx <= '0';

wait for 104160 ns;

Rx <= '0';

wait for 104160 ns;

Rx <= '0';

wait for 104160 ns;

Rx <= '1';

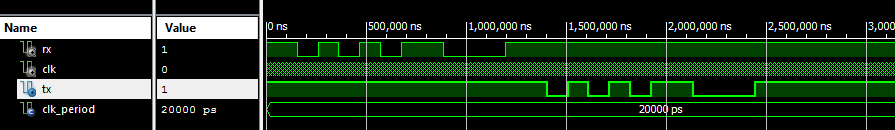
wait for 104160 ns;

Rx <= '1';

wait;

end process;

END;



**WEBGRAFÍA**

*ESTANDAR DE COMUNICACIONES INTERFAZ RS232 RS422 RS485 PUNTO FLOTANTE S.A.*. (2017). *Puntoflotante.net*. Recuperado el 05 de abril de 2017, de <http://www.puntoflotante.net/RS485.htm>

*ASCII*. (2017) (1st ed.). Madrid. Recuperado de <http://arantxa.ii.uam.es/~ig/practicas/enunciados/prac3/ascii.pdf>

*COMUNICACION POR PROTOCOLO RS-232*. (2017). *http://extropynow.weebly.com/*. Recuperado el 05 de abril de 2017, de <http://extropynow.weebly.com/uploads/1/6/4/1/16411724/mikroc-u_i-rs232-protocol.pdf>

Gómez Osuna, C., & López Vallejo, M. (2017). *Ejercicio 7*. *Laboratorio de Sistemas Integrados UPM*. Recuperado el 05 de abril de 2017, de <http://www.lsi.die.upm.es/~marisa/Teaching/LCSE/docs/Ejercicio_7-v3.pdf>

Méndez Ortiz, W., & Nieto Cubides, C. (2017). *Diseño de módulos en VHDL que permitan adicionar los periféricos PMB de Digilent al microprocesador embebido MicroBlaze, usando el sistema de desarrollo Spartan 3A DSP de Xilinx*. *Repositorio institucional Universidad Industrial de Santander*. Recuperado el 05 de abril de 2017, de <http://repositorio.uis.edu.co/jspui/bitstream/123456789/3414/2/132316.pdf>

*Norma RS232*. (2017). *Guillehg.com*. Recuperado el 05 de abril de 2017, de <http://www.guillehg.com/index.php?option=com_content&view=article&id=19&Itemid=656>

*La comunicación serie*. (2017). Recuperado el 05 de abril de 2017, de <http://perso.wanadoo.es/pictob/comserie.htm#la_norma_rs232>

Singh Dobal, N. (2013). Design of 4 Bit Serial IN - Parallel OUT Shift Register using Behavior Modeling Style (VHDL Code). *VHDL Programming*. Recuperado de <http://vhdlbynaresh.blogspot.com.co/2013/07/design-of-parallel-in-serial-out-shift.html>

*A UART Implementation in VHDL | "Domipheus Labs"*. (2017). *Labs.domipheus.com*. Recuperado el 05 de abril de 2017, de <http://labs.domipheus.com/blog/a-uart-implementation-in-vhdl/>

Singh Dobal, N. (2013). Design of Parallel In - Serial OUT Shift Register using Behavior Modeling Style (VHDL Code). *VHDL Programming*. Recuperado de <http://vhdlbynaresh.blogspot.com.co/2013/07/design-of-parallel-in-serial-out-shift.html>

Garzón,. (2017). *Timing Hazards (Retardos en Compuertas)*. *Es.slideshare.net*. Recuperado el 05 de abril de 2017, de <https://es.slideshare.net/junior198619/timing-hazards-retardos-en-compuertas>