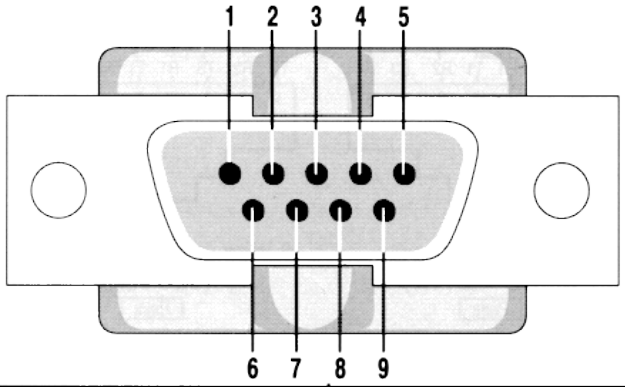
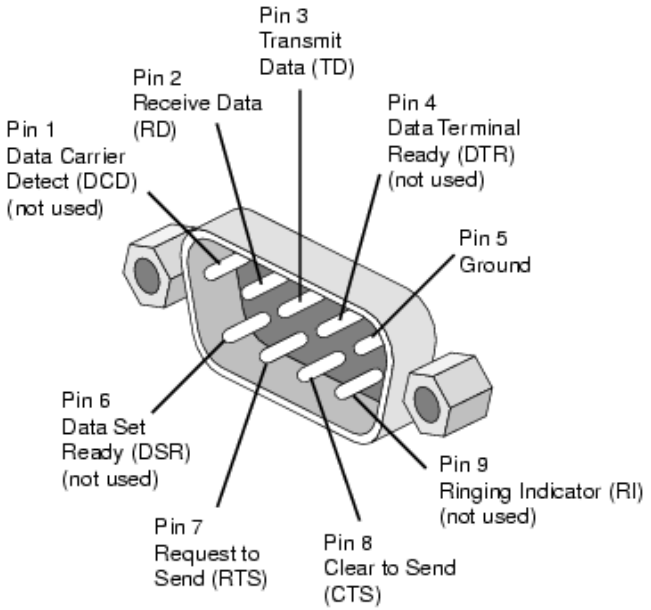
RS232 Port Interfacing Circuit

RS232 Port Pins:



Pin	Signal	Pin	Signal
1	Data Carrier Detect	6	Data Set Ready
2	Received Data	7	Request to Send
3	Transmitted Data	8	Clear to Send
4	Data Terminal Ready	9	Ring Indicator
5	Signal Ground		



RS232 is a serial communication port using -15V ... -3V for logic High and 3V ... 15V for logic Low. To communicate it with TTL devices such as an FPGA we need to use a serial level converter circuit.

The figures above show the arrangement of RS232 port pins. Pins that will be taken into account for this assignment are 2(rx), 3(tx) and 5(gnd).

Design Structure:

ASCII character is sent to RS232 port by Hyperterminal

8-bit ASCII character is sent to RS232 port

ASCII out

8-bit ASCII character is sent to FPGA serially bit by bit

Rx

8-bit ASCII data is reassembled and processed in FPGA

ASCII in

Tx

8-bit data is received back by PC on Hyperterminal as an ASCII character

8-bit data processed in FPGA is sent back to RS232 port serially bit by bit

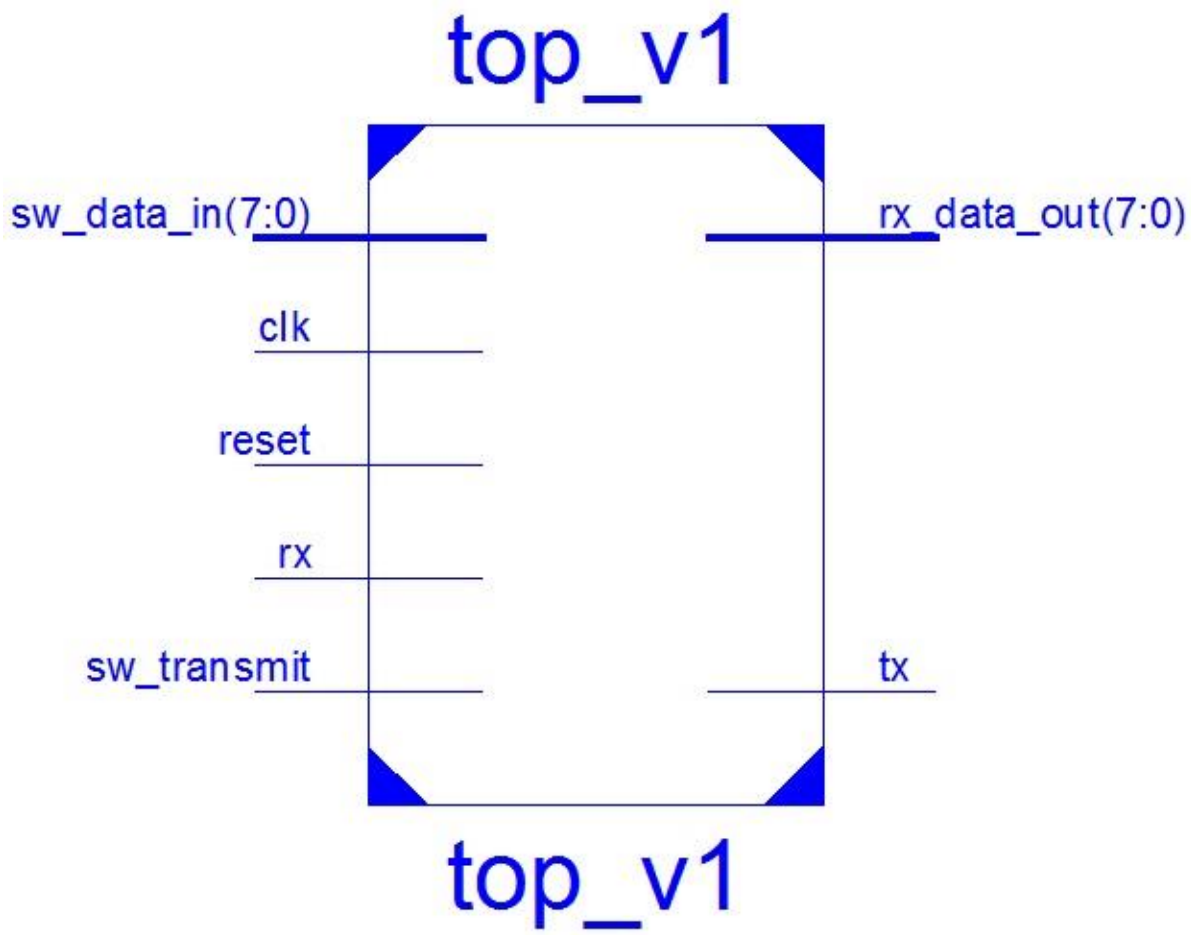
In this example Circuit:

- Hyperterminal sends ASCII character to FPGA through RS232 port serially.
- FPGA receives serial data and reassembles them inside.
- Reassembled data is processed in FPGA.
- Processed data is transmitted from RS232 port to Hyperterminal serially.
- Hyperterminal deserializes data and shows its ASCII equivalent.

You can download a 30 day trial version of Hyperterminal from the link below

http://www.hilgraeve.com/hyperterminal/

Top Level:


top_v1
sw_data_in(7:0)
clk
reset
rx
sw_transmit
rx_data_out(7:0)
tx
top_v1

Top Level:

Inputs:

clk : clock input supplied by FPGA.

reset : reset input tied to pushbutton BTN3 on FPGA.

rx : 1-bit data received from RS232 port to FPGA.

sw_transmit : pushbutton BTN0 that enables sending ASCII characters from switches to PC.

sw_data_in[7:0]: switches from which ASCII characters can be sent.

Outputs:

rx_data_out : led outputs that are connected to received 8-bit data.

tx : 1-bit data sent to RS232 port from FPGA.

Design Submodules:

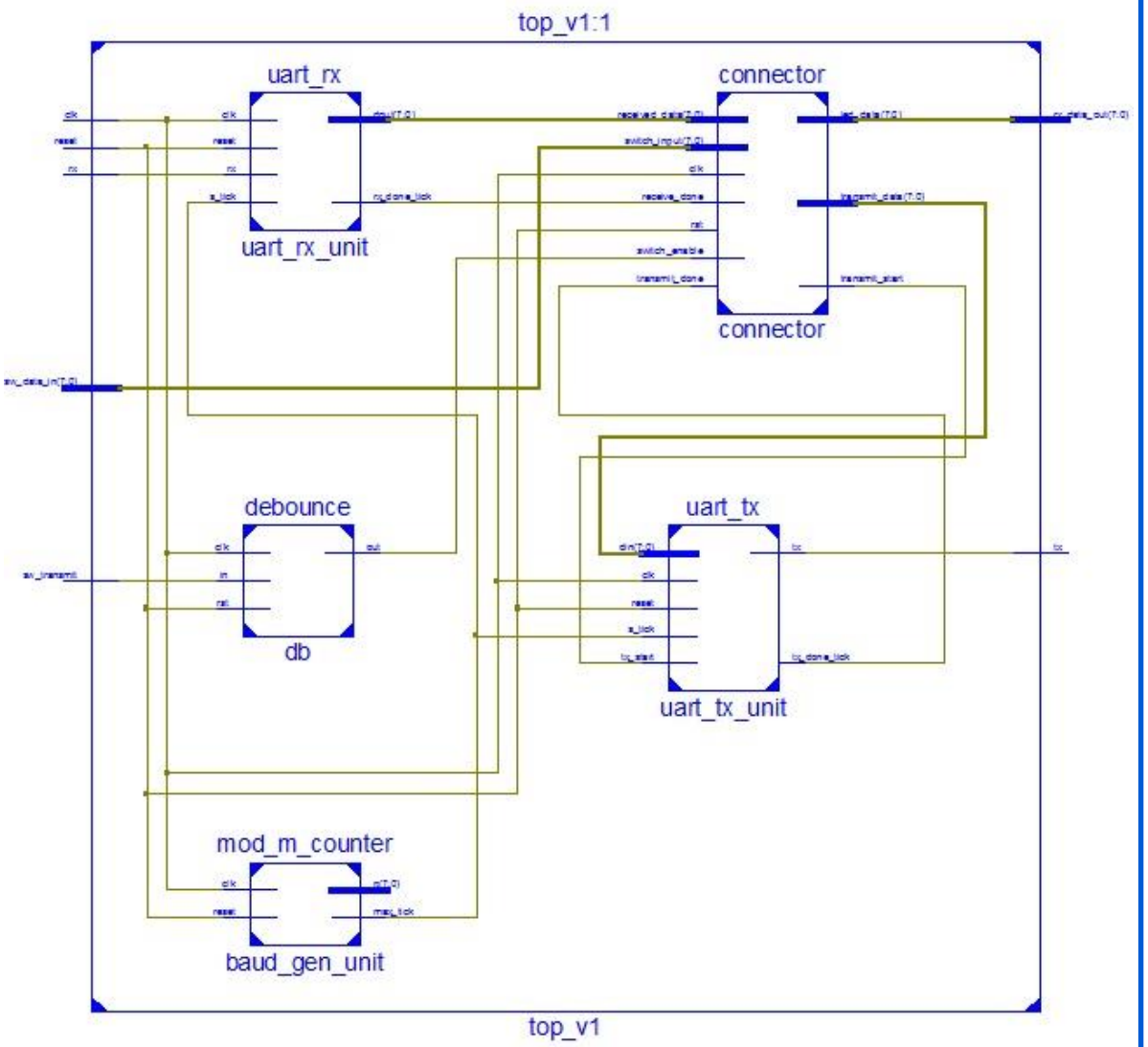
uart_rx: samples serial rx inputs and forms them into 8-bit received data.

uart_tx: takes 8-bit data to be sent and sends it serially.

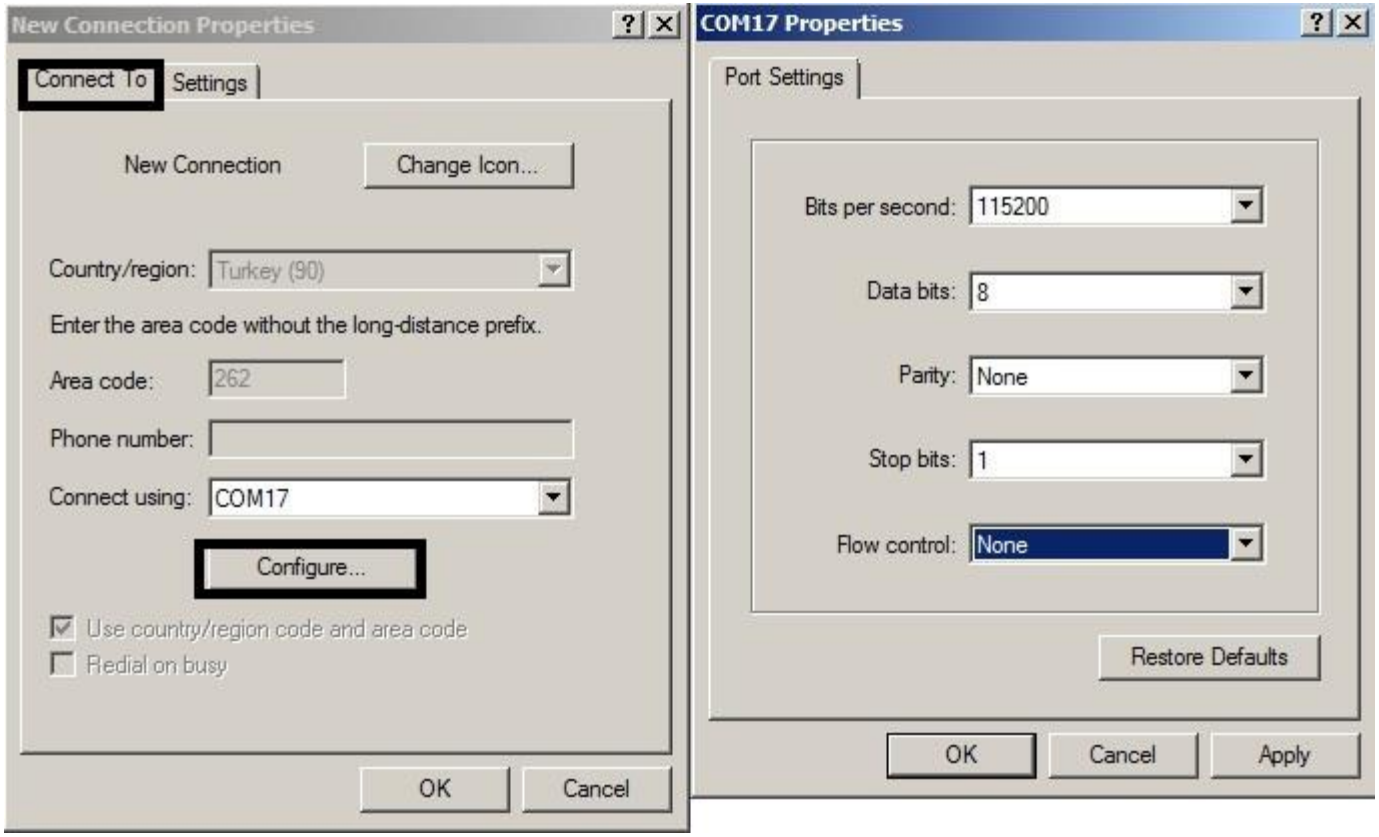
mod_m_counter: handles sampling rates.

debounce: debounces necessary switch and/or button signals.

connector: is the application module that will be modified. Default behavior is sending 'received data + 1' back.



Hyperterminal Settings:



From '*File -> Properties*' you should make the changes shown here and the next page.

Hyperterminal Settings: