Exercise 3: Serial Interface (RS232)

G. Kemnitz[*], TU Clausthal, Institute of Computer Science

May 23, 2012

Abstract

A working circuit design for the receiver of a serial interface is given. It has to be examined and tested. Afterward the corresponding transmitter has to be designed and the receiver has to be extended by additional functions.

1 RS232

The PCs in the exercise room and the experimental board are linked by a serial cable through which data may be transmitted in byte. From the 9-pin SUBD-plug of a serial terminal only 3 pins are used:

Pin 2: RxD, input receiver

Pin 3: TxD, output transmitter

Pin 5: ground

A zero is represented by a negative and a one by a positive voltage. The conversion of the usual logic levels high and low to positive and negative voltages is done in a special circuit, at the experimental board in IC14. In the ucf-file the receiver input is named RxD and the transmitter output TxD.

The transmission is organized byte-wise. During transfer pause the transmitted signal is TxD='1'. Each data package starts with a start bit '0'. Then follow 8 data bits, an optional parity bit and a stop bit that is '1' (see figure 1). After the stop bit the start bit of the next transmission may follow or the transfer pauses. The parity bit is the modulo-2- or EXOR-sum of the data bits and is used for error-detection.

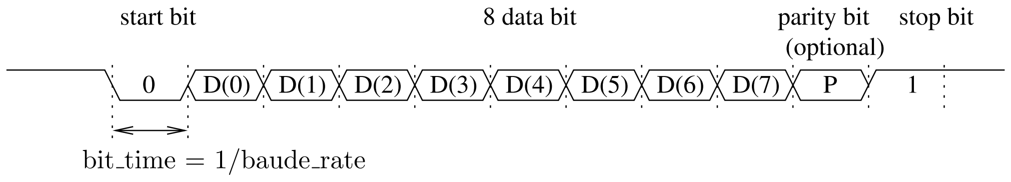


Figure 1: Data frame of an RS232 transmission

The transmitter of a serial terminal consists of

- a circuit to provide the clock and

[*]Tel. 05323/727116

- an automaton to transform a byte to a data sequence corresponding to the protocol.

The basic clock CLK on our experimental board has a frequency of 50MHz. This clock is usually first divided to a frequency 16 times higher then the baud rate and a duty cycle 1:1:

```
process(clk)
  variable counter : integer range 0 to cTeiler - 1 := cTeiler - 1;
begin
  if rising_edge(clk) then
    if counter = 0 then
      clk_16x_baud <= not clk_16x_baud;
      counter := cTeiler - 1;
    else
      counter := counter - 1;
    end if;
  end if;
end process;
```

The following tabular shows denominator values for various baud rates:

baud rate	cTeiler (basic clock 50 MHz)
2,4 kBaud	648
4,8 kBaud	324
9,6 kBaud	162
19,2 kBaud	81

The clock clk_16x_baud is divided by 16 in a subsequent divider (4 bit counter) and used as transmitter clock.

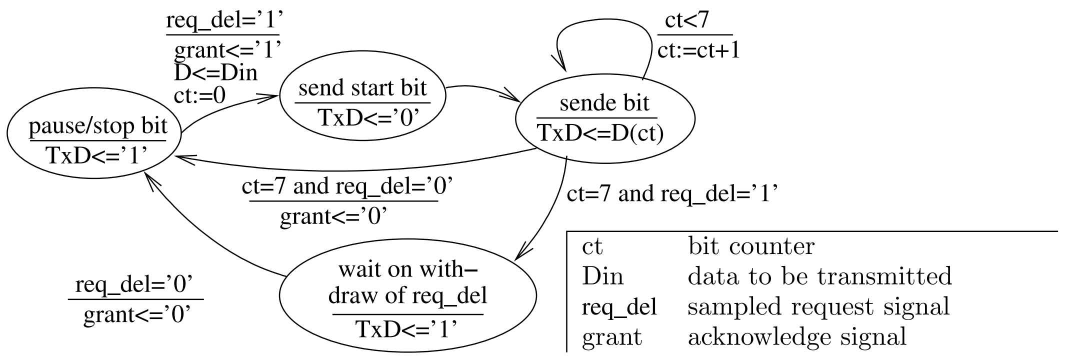
The transmitter automaton should be specified as a state graph. As long as there are no transmission, the automaton stays in the ≫pause≪ state and keeps the transmission line TxD = '1'. If a transmission starts the automaton goes through the states:

- send start bit: TxD<='0'
- send bit 0: TxD<=D(0)
- ...
- send bit 7: TxD<=D(7)
- send parity bit: TxD<=D(0) xor D(1) xor ... xor D(7)
- send stop bit (TxD<='1')

The data to be transmitted are generally produced by a circuit that is not clocked with the baud rate. To pass the data to the transmitter a handshake protocol is required (figure 2):

- the data source sets the request signal ≫req≪ to '1' and provide data at the bus ≫Din≪.
- the serial sender receives the data as soon as it is ready for reception and acknowledges receipt by grant<='1'.
- serial sender submits the data.
- serial sender waits on withdraw of the request (wait on req='0') and deactivates ≫grant≪.
- Waiting on the next request.

Send- and handshake sequence have to merged to one state graph, whereby only the sampled request signal should be processed (figure 3).



ct	bit counter
Din	data to be transmitted
req_del	sampled request signal
grant	acknowledge signal

Figure 2: State graph of the transmitter with handshake

Remarks:

- The automaton in figure transmits data without parity bit.
- The transmitter is described by an operation graph, which nodes control a counter and at which edges counter values are evaluated. The description also may be changed so, that the bit to be send is distinguished by the automaton state instead of the counter state.
- The data type of the counter signal should be an integer with a range from 0 to 7 to select one of 8 bits from ≫D≪.

The receiver works with the same baud rate as the transmitter. From this he knows the bit time. About the start time it is informed indirectly by the falling edge at the begin of the transmission. From start time and bit time the sample times for the single bits have to be calculated. The solution includes the 1:16 prescaler. The receiver starts as the transmitter with the pause state. Instead on a transmit requests it waits on a falling edge of the sampled receive signal ≫RxD_del≪. The signal ≫RxD≪ here has to be sampled with a several times higher rate, in the example the 16 times higher rate than the baud rate. As soon as the data line turns to zero, the automaton switches to the next state and sets the prescaler to zero. All further state transitions are performed when the prescaler reaches half of its maximum value, i.e. in the middle of the time between two state transitions of the sender, when the submitted data are stable and valid. During the first state transition a zero is expected on receiver line (start bit). With the 2. to 9. state transition data bits are sampled. During the 10th transition it will be checked that the stop bit is one. In case an additional parity bit is transferred for error detection, it also has to be sampled and checked.

The transfer of the received data to the following circuit generally is done via a handshake protocol (figure 3):

- after an error-free reception of a byte the serial receiver activates a write request signal
- waits on an acknowledge and deactivates the request signal
- waits on deactivation of the acknowledge signal.

Remarks:

- No parity bit.
- vt is a counter modulo 16.

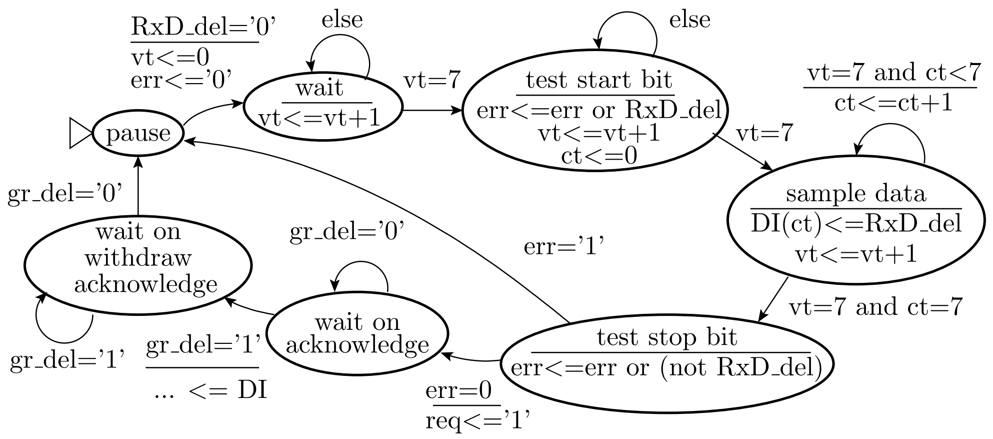


Figure 3: State graph of the receiver with handshake

- ct must have as for the transmitter the type integer range 0 to 7 to select 1 out of 8 bits in DI.
- The transmitter is described by an operation graph, which nodes control a counter and at which edges counter values are evaluated. The description also may be changed so, that the bit to be send is distinguished by the automaton state instead of the counter state.

2 Test of the example design

Download the files a3_rs232.xise, a3_rs232.vhd and a3_rs232.ucf from the web page in a project folder in your home account. Open the project and then the VHDL-file ≫a3_rs232.vhd≪ with ISE.

1. Reconstruct the state graph of the receiver from the file a3_rs232.vhd and draw it on paper. Compare the so found graph with figure 3.
   (a) Does it include a parity check?
   (b) What baud rate is configured?
   (c) Does the automaton include handshake?
   (d) How works th echo function?
2. Translate the design and program it into the FPGA.
3. Start a serial terminal on the PC, under Linux use ≫GtkTerm≪:
   (a) Anwendungen => Accessories => Serial port terminal
      - Select transmission parameters via the configuration menu ≫Port öffnen≪. Settings: Port: /dev/ttyS0, Speed: 9600 bits per second, Parity: even, Bits: 8, Stop bits: 1, Flow control: none

   Under Windows use ≫putty≪
   - ≫Start≪ ▷ ≫All Programs≪ ▷ ≫...≪ ▷ ≫Putty≪ ▷ select ≫serial ▷terminal settings via ≫Connection ▷ Serial≪ ▷ ≫Open≪
4. Test connection: Send after another the numbers 0 to 9 from the PC to the FPGA and check, that on led(7 downto 0) the corresponding ASCII-Codes 30h to 39h is displayed.

3 Design exercises

1. Plug the logic analyzer to the pins "DB0" (RxD) and "ADR0" (TxD) of the test point header module on expansion connector A2. Download from the web page the file ≫a3_rs232.xml≪. Adjust the sampling rate and the trigger condition for recording the serial signal from the PC. Why did you select this settings?

2. Design a send automaton, which on pressing button BTN0 sends byte SW(7 downto 0) to the PC. Attention, don't forget de bouncing. Write the send automaton in a new file. Use the following entity description:

```
entity a3_rs232_sender is
  port(
    clk_16x_baud : in  std_logic;
    txd          : out std_logic;
    data         : in  std_logic_vector(7 downto 0);
    req          : in  std_logic;
    grant        : out std_logic
  );
end entity;
```

   Simulate the automaton with the test-bench ≫a3_rs232_tb.vhd≪ from the web and test the circuit on the board. The logic analyzer may be helpful for trouble shooting.

3. Modify the receiver circuit so that on the two left 7-segment display digits the last received byte is displayed as two hexadecimal numbers. Add a parity check in stage 9 of the automaton and that possible errors are displayed at LED11 of the expansion board ≫traffic light control / number lock≪ at B1.

4 Check list for the compliance test

- State graphs of the send and the receive automaton on paper.
- presentation of the measurement results with the logic analyzer
- Simulation and presentation of the send automaton
- changes to the receiver.