```
----------------------------------------------------------------------------------
--
-- Company:
-- Engineer:
--
-- Create Date:    17:35:58 05/20/2012
-- Design Name:
-- Module Name:    send_block - Behavioral
-- Project Name:
-- Target Devices:
-- Tool versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------
--
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
USE IEEE.numeric_std.all;
use std.textio.all;
use ieee.std_logic_textio.all;

entity soft is
  generic (UART_Speed : integer := 115200;   -- UART Speed
           System_CLK : integer := 50000000;  -- System CLK in Hz
           comand_file : string
:="/home/Vidokq/djin18reader/vhd/D18/comfile.txt"
           );
  port (
    CLK      : out  std_logic;                     -- system clk
    RST_N    : out  std_logic;                     -- system reset#
    DATA_IN  : out  std_logic_vector(7 downto 0);  -- Transmit data
    DATA_OUT : in   std_logic_vector(7 downto 0);  -- Recieved data
    RX_VALID : in   std_logic;                     -- RX buffer data ready
    TX_VALID : out  std_logic;                     -- Data for TX avaible
    TX_BUSY  : in   std_logic;                     --
    RX_BUSY  : in   std_logic);

end soft;

architecture beh of soft is

  file comfile      : text open read_mode is comand_file;
  signal str        : string(1 to 1000) := (others => ' ');
--  signal ch         : character         := ' ';
  signal str_l      : natural;
  signal CLK_i      : std_logic         := '0';
  signal RST_N_i    : std_logic;
  signal DATA_IN_i  : std_logic_vector(7 downto 0);
  signal TX_VALID_i : std_logic;
  signal tmd : time := 0 ms;
  signal f_get_answer       : boolean := false;
  signal f_start_timer      : boolean := false;
  signal f_start_get_answer : boolean := false;
  signal f_timer            : boolean := false;
  signal f_byte_timer       : boolean := false;
  signal comand             : std_logic_vector(7 downto 0);

  procedure skip_space (variable l : inout line) is
```

```
    variable char : character;
    variable flag : boolean:=true;
  begin  -- skip_space
    while flag loop
      if l'length = 0 then
        flag := false;
      elsif l(1) = ' ' then
        read(l, char);
      else
        flag := false;
      end if;
    end loop;
  end skip_space;

begin  -- beh

wait_start_answer_timer: process
begin  -- process wiat_timer
  wait on f_start_timer;
  wait for 1 ms;
    f_timer <= f_start_timer and (not f_start_get_answer);
end process wait_start_answer_timer;


wait_byte_timer: process
begin  -- process wiat_timer
  wait for 0.5 ms;
  if (NOW - tmd) > 1 ms then
    if f_start_get_answer then
      f_byte_timer <= true;
      wait for 0.1 ms;
      f_byte_timer <= false;
    end if;
  end if;
end process wait_byte_timer;

p_get: process
    variable cnt_answer_byte  : integer := 0;
    variable num_answer_byte  : integer := 0;
    variable f_get_first_byte : boolean := false;
  begin  -- process p_get

    wait until RX_VALID = '1' or f_byte_timer;
    assert not f_byte_timer report "превышено время в одной посылке мужду
байтами 1мс" severity FAILURE;
    f_get_answer <= false;
    tmd <= NOW;

    case comand is
      when x"01"|x"02"|x"04" =>
        if cnt_answer_byte = 0 then
          assert (DATA_OUT = x"00") report "ошибка обрабоки данных код ошибки"
severity failure;
          f_get_answer <= true;
          wait for 1 us;
          f_get_answer <= false;
        end if;
      when x"03" =>
        if cnt_answer_byte = 0 then
          assert (DATA_OUT = x"00") report "ошибка обрабоки данных код ошибки"
severity failure;
          f_start_get_answer <= true;
          cnt_answer_byte   := cnt_answer_byte + 1;
        elsif cnt_answer_byte = 1 then
```

```
          num_answer_byte := to_integer(unsigned(DATA_OUT));
          cnt_answer_byte   := cnt_answer_byte + 1;
        elsif cnt_answer_byte = num_answer_byte + 1 then
          f_get_answer <= true;
          wait for 1 us;
          f_get_answer <= false;
          f_start_get_answer <= false;
          cnt_answer_byte   := 0;
        else
          cnt_answer_byte   := cnt_answer_byte + 1;
        end if;
      when others => null;
    end case;

  end process p_get;

  p_send      : process
   variable l : line;
   variable byte_r : std_logic_vector(0 to 7);
   variable good : boolean;
   variable f_first_byte : boolean:=false;
  begin  -- process p1

    loop1: while not endfile(comfile) loop

      readline (comfile,l);
      skip_space (l);                   -- пропускаем пробелы вначале строки
      next loop1 when l'length = 0;     -- читаем след. строку, если пустая
      -- копируем строку в сигнал

      for i in 1 to l'length loop
        str(i) <= l(i);
      end loop;  -- i
      str((l'length)+1 to 1000) <= (others => ' ');
      str_l <= l'length;

      if l(1) = '#' then                -- коментарий пропускаем
        assert false report "Коментарий:"&str(1 to str_l) severity NOTE;
      else                              -- данные
        assert false report "Посылаемые данные:"&str(1 to str_l) severity NOTE;

        f_first_byte := true;
        while l'length /= 0 loop
          hread (l,byte_r,good);
          assert good report "ошибка чтения строки (число символов должно быть
четным)" severity FAILURE;
          if f_first_byte then
            f_first_byte := false;      --сохраним первый байт
            comand <= byte_r;
          end if;
          TX_VALID_i <= '1';
          DATA_IN_i <= byte_r;
          skip_space (l);

          wait until TX_BUSY = '1';
        end loop;
        f_start_timer <= true;
        TX_VALID_i <= '0';              --во время ожидания ответа ничего не
посылаем
        wait until f_get_answer or f_timer;        --ждем пока придет ответ или
пройдет время
        assert not f_timer report "превышено время ожидания ответа 1 мс"
severity FAILURE;
        f_start_timer <= false;
```

```
      end if;
    end loop;

  assert false report "конец командного файла" severity failure;
  end process p_send;

  RST_N_i <= '0' , '1' after 100 ns;
  CLK_i <= not CLK_i after 1 ns * (1000000000 / System_CLK );
  RST_N <= RST_N_i;
  CLK <= CLK_i;
  DATA_IN <= DATA_IN_i;
  TX_VALID <= TX_VALID_i;
end beh;
```