EPP_DEVICE 1

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use std.textio.all;
use ieee.std_logic_textio.all;

entity epp_device1 is
  port (
    DATA   : inout std_logic_vector(7 downto 0);
    PWRITE : out   std_logic;
    DSTRB  : out   std_logic;
    ASTRB  : out   std_logic;
    PWAIT  : in    std_logic);

end epp_device1;
architecture sim of epp_device1 is
  constant T_clk_epp      : time      := 100 ns;  -- Internal clock period.
  signal   clk_epp        : std_logic := '0';     -- Internal clock signal.
  signal read_value : std_logic_vector(7 downto 0) := (others => '0');
  constant dir_frec : std_logic_vector( 7 downto 0) := x"F0";
  constant dir_dpram1 : std_logic_vector( 7 downto 0) := x"A1";
  constant dir_dpram2 : std_logic_vector( 7 downto 0) := x"A2";
  constant EPP_cicle_length       :       real := 0.5;
begin
  -- internal clock signal generation.

  clk_epp <= not(clk_epp) after T_clk_epp/2;



 process

    procedure epp_cicle ( address : in    std_logic_vector(7 downto 0);
                          data_io : inout std_logic_vector(7 downto 0);
                          r_w     : in    character) is

    begin
      wait until clk_epp = '1';
      PWRITE   <= '0';
      wait until clk_epp = '1';
      ASTRB    <= '0';
      data     <= address;
      wait for T_clk_epp*EPP_cicle_length;
      ASTRB    <= '1';
      wait until clk_epp = '1';
      data     <= (others => 'Z');
      PWRITE   <= '1';
      wait until clk_epp = '1';
      wait for T_clk_epp*EPP_cicle_length;
-------------------------------------------------------------------------
      if r_w = 'w' then                 --  write   cicle
```

```
        PWRITE <= '0';
        data   <= data_io;
      end if;
-------------------------------------------------------------------------------
      wait until clk_epp = '1';
      DSTRB     <= '0';
      wait for T_clk_epp*EPP_cicle_length;
      if r_w = 'r' then
        data_io:= data;
      end if;

      DSTRB   <= '1';
      wait until clk_epp = '1';
      data   <= (others => 'Z');
      PWRITE <= '1';
      wait until clk_epp = '1';

    end procedure;

file arch_in : text ;
 variable bf : line;
    variable dato_rd : std_logic_vector(7 downto 0);
     variable dato : std_logic_vector(7 downto 0);
    variable dir  : std_logic_vector(7 downto 0);
  begin
    --inicialización
    data   <= (others => 'Z');
    PWRITE <= '1';
    DSTRB   <= '1';
    ASTRB   <= '1';
    dir  := (others   => '0');
    wait for 160 ns;


    file_open(arch_in,"../Souce/Vo1.dat",read_mode);
     dir:= dir_dpram1        ;
    while not endfile(arch_in) loop
      readline (arch_in, bf);
      hread (bf, dato_rd);
      dato:=dato_rd;
      epp_cicle ( address => dir,
                  data_io => dato,
                  r_w     => 'w');

    end loop;
    file_close(arch_in);

 file_open(arch_in,"../Souce/Vo2.dat",read_mode);
    dir:= dir_dpram2        ;

    while not endfile(arch_in) loop
      readline (arch_in, bf);
```

```
      hread (bf, dato_rd);
      dato:=dato_rd;
      epp_cicle ( address => dir,
                  data_io => dato,
                  r_w     => 'w');
    end loop;
    file_close(arch_in);

    dir := dir_frec ;
          dato:=x"13";
    epp_cicle ( address => dir,
                  data_io       => dato,
                  r_w           => 'w');

wait for 5 ms;

    dir := dir_frec ;
          dato:=x"73";
    epp_cicle ( address => dir,
                  data_io       => dato,
                  r_w           => 'w');


wait for 5 ms;

          dato:=x"bf";
    epp_cicle ( address => dir,
                  data_io       => dato,
                  r_w           => 'w');

    wait for 5 ms;

          dato:=x"ff";
    epp_cicle ( address => dir,
                  data_io       => dato,
                  r_w           => 'w');


wait for 5 ms;
   report "FIN CONTROLADO DE LA SIMULACION" severity failure;
  end process;
end sim;
```