```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;

entity controllore is
  port (
    CLK       : in  std_logic;
    RESET     : in  std_logic;
    SERIALINE : in  std_logic;
    PASSWORD  : in  unsigned(9 downto 0);
    ERROR_P   : out std_logic;
    ERROR_PW  : out std_logic;
    CARD_OUT  : out std_logic;
    ID_USER   : out unsigned(11 downto 0);
    OK        : out std_logic;
    CODE_REQ  : out std_logic);
end controllore;


architecture A of controllore is
  type stato  is (idle,start,B0,B1,B2,B3,BP,campiona,confronta,
                  erroreP,errorePW,stop,rilascia,pw_wait,pw_req,corretto);
  signal present_state,next_state : stato;
  signal c_count,n_count : unsigned(3 downto 0);
  signal RESET_t, reset_timer,reset_reg,res_reg : std_logic;
  signal parity1,parity2,parity3,pcheck : std_logic;
  signal fineciclo, en_timer,campionaTbit  : std_logic;
  signal q : unsigned(5 downto 0);
  signal dato1,dato2,dato3 : unsigned(4 downto 0);
  signal enpar1,enpar2,enpar3,enpar : std_logic;
  signal cpar_count,npar_count : unsigned(1 downto 0);
  signal identif : unsigned(11 downto 0);
  signal check_A,check_B,check_C,check : std_logic;
  signal n_count_5s,c_count_5s : unsigned(15 downto 0);
  signal reset_5s,res_5stimer,s5 : std_logic;
  signal en_shift : std_logic;


begin

-- circuito della porta seriale --

  -- contatore dei cicli di clock
  -- necessari per scandire il Tbit
  -- 1Tbit = 1/1000 sec => 10 cicli di clock
  -- contatore a 4 bit
  process (CLK, reset_t)
  begin
    if RESET_t = '1' then
      c_count <= (others => '0');
    elsif CLK'event and CLK = '1' then  -- rising clock edge
      if en_timer = '1' then
        c_count <= n_count;
      end if;
    end if;
  end process;

  n_count <= c_count + 1;
  RESET_t <= RESET or fineciclo;
  fineciclo <= '1' when c_count = conv_unsigned(9,4) else '0';
```

```
---- registro  a scorrimento
  shift_reg0: for I in 0 to 4 generate

    ff_I : process(CLK)
    begin
      if CLK'event and CLK='1' then
        if RESET='1' then
          q(I+1) <= '0';
        elsif campionaTbit='1' then
          q(I+1) <= q(I);
        end if;
      end if;
    end process;

  end generate;

  q(0) <= serialine;
  campionaTbit <= '1' when (c_count = 4 and en_shift = '1') else '0';

----registri per campionare il dato ricevuto (ho bisogno di un contatore per
  --decidere su quale campionare)
  process (CLK, reset_reg)
  begin
    if RESET_reg = '1' then
      dato1 <= (others => '0');
    elsif CLK'event and CLK = '1' then  -- rising clock edge
      if enpar1 = '1' then
        dato1 <= q(5 downto 1);
      end if;
    end if;
  end process;

  process (CLK, reset_reg)
  begin
    if RESET_reg = '1' then
      dato2 <= (others => '0');
    elsif CLK'event and CLK = '1' then  -- rising clock edge
      if enpar2 = '1' then
        dato2 <= q(5 downto 1);
      end if;
    end if;
  end process;

  process (CLK, reset_reg)
  begin
    if RESET_reg = '1' then
      dato3 <= (others => '0');
    elsif CLK'event and CLK = '1' then  -- rising clock edge
      if enpar3 = '1' then
        dato3 <= q(5 downto 1);
      end if;
    end if;
  end process;

  process (CLK, reset_reg)
  begin
    if RESET_reg = '1' then
      cpar_count <= (others => '0');
    elsif CLK'event and CLK = '1' then  -- rising clock edge
      if enpar = '1' then
```

```
        cpar_count <= npar_count;
      end if;
    end if;
  end process;

  npar_count <= cpar_count + 1;
  enpar1 <= '1' when (enpar = '1' and cpar_count = 0) else '0';
  enpar2 <= '1' when (enpar = '1' and cpar_count = 1) else '0';
  enpar3 <= '1' when (enpar = '1' and cpar_count = 2) else '0';

  reset_reg <= RESET or res_reg;

  identif <= dato3(3 downto 0) & dato2(3 downto 0) & dato1(3 downto 0);
  ID_USER <= identif;

  --rete logica che calcola i bit di parità e
  --li confronta con i ricevuti. Poichè non mi interessa sapere
  --su quale dato si è verificato un errore mi basta un semplice
--albero di EXOR  ed assicurarmi di resettare a zero i registri qualora si
--verificasse un errore (cmq anche con un contatore va bene)
  parity1 <= (dato1(3) xor  dato1(2)) xor  (dato1(1) xor  dato1(0));
  parity2 <= (dato2(3) xor  dato2(2)) xor  (dato2(1) xor  dato2(0));
  parity3 <= (dato3(3) xor  dato3(2)) xor  (dato3(1) xor  dato3(0));

  process(parity1,parity2,parity3,dato1,dato2,dato3)
  begin
    if ((parity1 /= dato1(4)) or (parity2 /= dato2(4)) or (parity3 /= dato3(4))) then
      pcheck <= '1';
    else
      pcheck <= '0';
    end if;
  end process;

  --pcheck <= '1' when ((parity1 /= dato1(4)) or (parity2 /= dato2(4)) or (parity3 /=
dato3(4))) else '0';

  -- parte sequenziale della FSM
  process(clk)
  begin
    if clk'event and clk = '1' then
      if reset = '1' then
        present_state <= idle;
      else
        present_state <= next_state;
      end if;
    end if;
  end process;

  -- processo combinatorio, calcolo uscite e stato successivo
  process(present_state, SERIALINE,fineciclo,pcheck,cpar_count,s5,Password,check)
  begin

    en_timer <= '0';
    en_shift <= '0';
    res_5stimer <= '1';
    CARD_OUT <= '0';
    enpar <= '0';
    OK <= '0';
    error_PW <= '0';
    error_P <= '0';
    CODE_REQ <= '0';
```

```
    res_reg <= '0';
    case present_state is

      when idle =>
        if SERIALINE  = '0' then
          next_state <= start;
        else
          next_state <= idle;
        end if;

      when start => --aspetto 10 cicli di clock
        en_timer <= '1';
        en_shift <= '0';
        if fineciclo  = '1' then
          next_state <= B0;
        else
          next_state <= start;
        end if;

      when B0 => --aspetto 10 cicli di clock e campiono al quinto
        en_timer <= '1';
        en_shift <= '1';
        if fineciclo  = '1' then
          next_state <= B1;
        else
          next_state <= B0;
        end if;

      when B1 =>
        en_timer <= '1';
        en_shift <= '1';
        if fineciclo  = '1' then
          next_state <= B2;
        else
          next_state <= B1;
        end if;

      when B2 =>
        en_timer <= '1';
        en_shift <= '1';
        if fineciclo  = '1' then
          next_state <= B3;
        else
          next_state <= B2;
        end if;

      when B3 =>
        en_timer <= '1';
        en_shift <= '1';
        if fineciclo  = '1' then
          next_state <= BP;
        else
          next_state <= B3;
        end if;

      when BP =>
        en_timer <= '1';
        en_shift <= '1';
        if fineciclo  = '1' then
          next_state <= campiona;
        else
```

```
      next_state <= B3;
    end if;

  when campiona =>
    en_timer <= '1';
    enpar <= '1';
    next_state <= stop;

  when stop =>
    if fineciclo  = '1' then
      next_state <= confronta;
    else
      next_state <= stop;
    end if;

  when confronta  =>
    if pcheck = '1' then
      next_state <= erroreP;
    elsif cpar_count = conv_unsigned(3,2) then --tutti i dati trasmessi
      next_state <= pw_req;
    else -- nuova lettura
      next_state <= idle;
    end if;

  when erroreP =>
    error_P <='1';
    next_state<= rilascia;

  when rilascia =>
    res_reg <= '1';
    CARD_OUT <= '1';
    res_5stimer <= '0';
    if s5 = '1' then
      next_state <= idle;
    else
      next_state <=present_state;
    end if;

  when pw_req =>
    CODE_REQ <= '1';
    next_state <= pw_wait;

  when pw_wait =>
    res_5stimer <= '0';

    if s5 = '0' then
      if PASSWORD /= conv_unsigned(0,12) then
        if check = '1' then
          next_state<= corretto;
        else
          next_state <= errorePW;
        end if;
      else next_state <= present_state;
      end if;
    else
      next_state <= errorePW;
    end if;

  when errorePW =>
    error_PW <='1';
    next_state<= rilascia;
```

```
      when corretto =>
        OK <= '1';
        res_reg <= '1';

        next_state <= idle;

      when others =>
        next_state <= idle;

    end case;
  end process;

--parte 2
  check_A <= '1' when ( identif = conv_unsigned(16#12A#,12) and PASSWORD = conv_unsigned
(128,10)) else '0';
  check_B <= '1' when ( identif = conv_unsigned(16#34B#,12) and PASSWORD = conv_unsigned
(256,10)) else '0';
  check_C <= '1' when ( identif = conv_unsigned(16#56C#,12) and PASSWORD = conv_unsigned
(512,10)) else '0';
  check <= check_A or check_B or check_C;

-- contatore 5 secondi -> 50000 cicli -> 16 bit
  process (CLK, reset_5s)
  begin
    if RESET_5s = '1' then
      c_count_5s <= (others => '0');
    elsif CLK'event and CLK = '1' then  -- rising clock edge
      c_count_5s <= n_count_5s;
    end if;
  end process;

  n_count_5s <= c_count_5s + 1;
  RESET_5s <= RESET or res_5stimer;
  s5 <= '1' when c_count_5s = conv_unsigned(5,16) else '0';
-- s5 <= '1' when c_count_5s = conv_unsigned(50000,16) else '0';

end  A;
```