```
--*************************************************************************************************
-- Product:  EZ2SUSB
-- Filename: sample1.vhd
-- Date:     08-28-2003

-- EasyFPGA
-- 905 Shell Blvd.,#202N
-- Foster City, CA 94404
-- USA
-- support@easyfpga.com
-- If this code works then it was written by Peter Bolech. If not, I don't know who wrote it.
--*************************************************************************************************
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity sample1 is port (

  -- Clock and reset --
  LCLK:                 in std_logic;                       -- clock 10 MHz, on-board clock
  nRESET:               in std_logic;                       -- FPGA reset, on-board reset

  -- FT245BM USB FIFO interface --
  USB_D:                inout std_logic_vector(7 downto 0);  -- FT245BM data bus
  nUSB_RD:              inout std_logic;                     -- FT245BM read enable
  USB_WR:               inout std_logic;                     -- FT245BM write enable
  nUSB_TXE:             in std_logic;                        -- FT245BM transmit FIFO empty flag
  nUSB_RXF:             in std_logic;                        -- FT245BM receive FIFO full flag

  -- LEDs
  LED:                  out std_logic_vector(7 downto 0);    -- on-board LEDs

  -- I/O headers
  IO:                   inout std_logic_vector(31 downto 0); -- I/O pins

  -- EXP header
  EXT_CLOCK_IN0:        in std_logic;                        -- external clock/user input
  EXT_CLOCK_IN1:        in std_logic;                        -- external clock/user input
  EXT_CLOCK_IN2:        in std_logic;                        -- external clock/user input
  EXP_IO:               inout std_logic_vector(24 downto 0)  -- I/O pins

   );
  end sample1;
```

```
architecture sample1_arch of sample1 is

signal COUNTER:          std_logic_vector(2 downto 0);
signal COUNT_ENAB:       std_logic;
signal PRESCALER:        std_logic_vector(21 downto 0);
signal VALUE:            std_logic_vector(21 downto 0);
signal REG:              std_logic_vector(7 downto 0);

begin

IO <= (others => 'Z');
EXP_IO <= (others => 'Z');

--*************************************************************************************************
--
--
USB_RD_PR: process(LCLK)
begin

if(LCLK'event and LCLK = '1') then
 if(nUSB_RD = '0') then
  nUSB_RD <= '1';
 elsif(nUSB_RXF = '0') then
  nUSB_RD <= '0';
 else
  nUSB_RD <= '1';
 end if;
end if;

end process USB_RD_PR;

--*************************************************************************************************
--
--
USB_WR_PR: process(LCLK)
begin

if(LCLK'event and LCLK = '1') then
  USB_WR <= '0';
end if;

end process USB_WR_PR;
```

```
--*************************************************************************************
-- access to the register
--
USB_REG_WRITE_PR: process(LCLK)
begin

 if(LCLK'event and LCLK = '1') then
  if(nUSB_RD = '0') then
   REG <= USB_D;
  end if;
 end if;

 USB_D <= (others => 'Z');

end process USB_REG_WRITE_PR;

--*************************************************************************************
-- Walking-one
--
LED_PR: process(LCLK)
begin

 if(LCLK'event and LCLK = '1') then
  if(nRESET = '0') then
   VALUE <= "00011110100001001000000";   -- 50 ms
  elsif(REG = "00000000") then
   VALUE <= "00011110100001001000000";   -- 50 ms
  elsif(REG = "00000001") then
   VALUE <= "00111101000010010000000";   -- 100 ms
  elsif(REG = "00000010") then
   VALUE <= "01111010000100100000000";   -- 200 ms
  end if;
 end if;

 -- prescaler counter
 if(LCLK'event and LCLK = '1') then
  if(nRESET = '0') then
   PRESCALER <= (others => '0');
  elsif(PRESCALER = VALUE) then
   PRESCALER <= (others => '0');
  else
   PRESCALER <= PRESCALER + '1';
  end if;
 end if;
```

```
-- clock enable
 if(LCLK'event and LCLK = '1') then
  if(PRESCALER = VALUE) then
   COUNT_ENAB <= '1';
  else
   COUNT_ENAB <= '0';
  end if;
 end if;

-- 3 bit binary counter: 0, 1, 2, 3, 4, 5, 6, 7
 if(LCLK'event and LCLK = '1') then
  if(nRESET = '0') then
   COUNTER <= (others => '0');
  elsif(COUNT_ENAB = '1') then
   COUNTER <= COUNTER + '1';
  end if;
 end if;

-- mux
 if(LCLK'event and LCLK = '1') then
  case COUNTER is
   when "000" =>
    LED <= "00000001";  -- LED 1
   when "001" =>
    LED <= "00000010";  -- LED 2
   when "010" =>
    LED <= "00000100";  -- LED 3
   when "011" =>
    LED <= "00001000";  -- LED 4
   when "100" =>
    LED <= "10000000";  -- LED 8
   when "101" =>
    LED <= "01000000";  -- LED 7
   when "110" =>
    LED <= "00100000";  -- LED 6
   when "111" =>
    LED <= "00010000";  -- LED 5
   when others =>
    LED <= "00000000";  -- remove latch !!!
  end case;
 end if;

end process LED_PR;

end sample1_arch;
```