```
--
-- TeC7 VHDL Source Code
--    Tokuyama kousen Educational Computer Ver.7
--
-- Copyright (C) 2002-2011 by
--                      Dept. of Computer Science and Electronic Engineering,
--                      Tokuyama College of Technology, JAPAN
--
--   上記著作権者は，Free Software Foundation によって公開されている GNU 一般公
-- 衆利用許諾契約書バージョン２に記述されている条件を満たす場合に限り，本ソース
-- コード(本ソースコードを改変したものを含む．以下同様)を使用・複製・改変・再配
-- 布することを無償で許諾する．
--
--   本ソースコードは＊全くの無保証＊で提供されるものである。上記著作権者および
-- 関連機関・個人は本ソースコードに関して，その適用可能性も含めて，いかなる保証
-- も行わない．また，本ソースコードの利用により直接的または間接的に生じたいかな
-- る損害に関しても，その責任を負わない．
--
--
--  TeC Panel
--

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

entity TEC_PANEL is
  port ( P_CLK      : in  std_logic;                      -- Clock
         P_2_4kHz   : in  std_logic;                      -- 2.4kHz
         P_18_75Hz  : in  std_logic;                      -- 18.75Hz
         P_2_3Hz    : in  std_logic;                      -- 2.3Hz
         P_RESET    : out std_logic;                      -- Reset Out Put
         P_AIN      : in  std_logic_vector(7 downto 0);   -- ADDR BUS
         P_LI       : in  std_logic;                      -- Instruction Fetch
         P_HL       : in  std_logic;                      -- Halt Request
         P_ER       : in  std_logic;                      -- Error
         P_MR       : in  std_logic;                      -- Memory Request
         P_STOP     : out std_logic;                      -- Stop

         P_RESET_SW : in  std_logic;                      -- Reset SW
         P_DATA_SW  : in  std_logic_vector(7 downto 0);   -- Data  SW
         P_SETA_SW  : in  std_logic;                      -- SETA  SW
         P_INCA_SW  : in  std_logic;                      -- INCA  SW
         P_DECA_SW  : in  std_logic;                      -- DECA  SW
         P_WRITE_SW : in  std_logic;                      -- WRITE SW
         P_STEP_SW  : in  std_logic;                      -- STEP  SW
         P_BREAK_SW : in  std_logic;                      -- BREAK SW
         P_STOP_SW  : in  std_logic;                      -- STOP  SW
         P_RUN_SW   : in  std_logic;                      -- RUN   SW
         P_RCW_SW   : in  std_logic;                      -- Rotate SW(CW)
         P_RCCW_SW  : in  std_logic;                      -- Rotate SW(CCW)

         P_R_LED    : out std_logic;                      -- Run LED
         P_SPK      : out std_logic;                      -- 操作音の出力
         P_A_LED    : out std_logic_vector(7 downto 0);   -- Address LED
         P_SEL      : out std_logic_vector(2 downto 0);   -- Rotate SW(Output)
         P_WRITE    : out std_logic;                      -- WRITEスイッチの操作
         P_INT      : out std_logic                       -- Interrupt SW
       );
end TEC_PANEL;


architecture RTL of TEC_PANEL is

-- トリガ付きのスイッチ
component TRSW
```

```
  port ( P_CLK    : in  std_logic;                        -- CLK
         P_RESET  : in  std_logic;                        -- Reset
         P_S      : in  std_logic;                        -- S
         P_SMP    : in  std_logic;                        -- Sample
         P_RPT    : in  std_logic;                        -- Repeate
         P_Q      : out std_logic                         -- Q
       );
end component;

-- レジスタ
signal I_ADR     : std_logic_vector(7 downto 0);        -- アドレスレジスタ

-- FF
signal I_SSFF    : std_logic;                            -- Start/Stop FF
signal I_ERROR   : std_logic;                            -- Error FF

-- LEDの点滅速度やスイッチサンプリングタイミングを決めるタイマ
signal I_SMP     : std_logic;                            -- 18.75Hz のトリガ
signal I_18_75Hz : std_logic;                            -- パルス生成用

-- ロータリースイッチの内部状態と出力
signal I_SW    : std_logic_vector(2  downto 0);
signal I_MM    : std_logic;

-- 操作音を出すため
signal I_CLICK  : std_logic;      -- 操作音を鳴らすイベントが発生した
signal I_BEEP   : std_logic;      -- イベントを I_SMP まで保留
signal I_BEEP_D : std_logic;      -- I_SMP 1周期の間、ブザーを鳴らす
signal I_ROT    : std_logic;      -- ロータリースイッチが回転した
signal I_RESET_D: std_logic;      -- RESETでトリガを作るため
signal I_SETA_D : std_logic;      -- SETAでトリガを作るため
signal I_STOP_D : std_logic;      -- STOPでトリガを作るため

-- スイッチの入力をトリガまたはクロックに同期した信号
signal I_RCW   : std_logic;
signal I_RCCW  : std_logic;
signal I_INCA  : std_logic;
signal I_DECA  : std_logic;
signal I_WRITE : std_logic;
signal I_RUN   : std_logic;
signal I_SETA  : std_logic;
signal I_STOP  : std_logic;
signal I_RESET : std_logic;

signal I_VOL   : std_logic;                    -- 操作音の大きさ(1:大 / 0:小)

-- 定数用
signal logic0  : std_logic;
signal logic1  : std_logic;

begin
  logic0 <= '0';
  logic1 <= '1';

  -- スイッチサンプリングタイミング用タイマ
  process(P_CLK)
  begin
    if (P_CLK'event and P_CLK='1') then
      if (P_18_75Hz='0' and I_18_75Hz='1') then  -- エッジの検出
        I_SMP <= '1';                            -- サンプリング用のパルスを生成
      else
        I_SMP <= '0';
      end if;
      I_18_75Hz <= P_18_75Hz;                    -- エッジ検出用の1クロック遅れ
信号
    end if;
```

```
end process;

-- スイッチの操作音の制御   --MM の場合だけ操作可能に変更(2008.7.10)--
I_CLICK <= (I_INCA and I_MM)
          or (I_DECA and I_MM)
          or (I_ROT or not I_RESET)
          or (I_WRITE and not I_SSFF )
          or (I_SETA  and not I_SETA_D and I_MM)
          or (I_STOP  and not I_STOP_D and I_SSFF)
          or (I_STOP  and not I_STOP_D and I_ERROR)
          or (I_RUN   and not I_SSFF);

-- 操作音を作る、通常(I_VOL='0')なら「ピッ！」の音、そうでなければ「プチ」の音
P_SPK <= (I_VOL or P_2_4kHz) and I_BEEP_D;

-- 操作音の継続時間を一定に保つ (1/18.75秒にする)
process(P_CLK)
begin
  if (P_CLK'event and P_CLK='1') then
       if (I_CLICK='1') then
            I_BEEP   <= '1';
       elsif (I_SMP='1') then
            I_BEEP_D <= I_BEEP;
            I_BEEP   <= '0';
       end if;
   end if;
end process;

-- リセットスイッチ(リピートさせない)
-- パワーオンリセットに失敗することがあるので、
-- 長いパルスを出力するように変更  2004.7.16
P_RESET <= I_RESET;
process(P_CLK)
begin
  if (P_CLK'event and P_CLK='1') then
      if (I_SMP='1') then
          if (P_RESET_SW='1' and I_RESET_D='0') then
              I_RESET   <= '0';
              I_RESET_D <= '1';
              if (P_STOP_SW='1' and P_SETA_SW='0') then
                  I_VOL <= not I_VOL;
              end if;
          else
            I_RESET   <= '1';
          end if;
          if (P_RESET_SW='0') then
              I_RESET_D <= '0';
          end if;
      end if;
   end if;
end process;

-- トリガ付きスイッチ
INCA_SW  :TRSW port map(P_CLK, I_RESET, P_INCA_SW,  I_SMP, logic1, I_INCA  );
DECA_SW  :TRSW port map(P_CLK, I_RESET, P_DECA_SW,  I_SMP, logic1, I_DECA  );
WRITE_SW :TRSW port map(P_CLK, I_RESET, P_WRITE_SW, I_SMP, I_MM,   I_WRITE );
RUN_SW   :TRSW port map(P_CLK, I_RESET, P_RUN_SW,   I_SMP, logic0, I_RUN   );
RCW_SW   :TRSW port map(P_CLK, I_RESET, P_RCW_SW,   I_SMP, logic1, I_RCW   );
RCCW_SW  :TRSW port map(P_CLK, I_RESET, P_RCCW_SW,  I_SMP, logic1, I_RCCW  );

-- STOP, SETA スイッチ(リピートしても良い)
process(P_CLK)
begin
  if (P_CLK'event and P_CLK='1') then
    if (I_SMP='1') then
      I_SETA  <= P_SETA_SW;
```

```
        I_STOP  <= P_STOP_SW;
      end if;
      I_SETA_D  <= I_SETA;
      I_STOP_D  <= I_STOP;
    end if;
  end process;

  -- ロータリースイッチの代替
  I_MM  <= I_SW(2) and not I_SW(1) and I_SW(0);
  P_SEL <= I_SW;
  process(P_CLK)
  begin
    if (P_CLK'event and P_CLK='1') then
      if (I_RCW='1' and not (I_SW="101")) then
        I_SW  <= I_SW + 1;
        I_ROT <= '1';
      elsif (I_RCCW='1' and not (I_SW="000")) then
        I_SW  <= I_SW - 1;
        I_ROT <= '1';
      else
        I_ROT <= '0';
      end if;
     end if;
  end process;

  -- パネルから発生する書き込み信号
  P_WRITE  <= not I_SSFF and I_WRITE;

  -- パネルから発生する割り込み
  P_INT    <=     I_SSFF and I_WRITE;

  -- アドレスレジスタ --MM の場合だけ操作可能に変更(2008.7.10)--
  P_A_LED  <= I_ADR when I_MM='1' else "00000000";
  process(P_CLK, I_RESET)
  begin
    if (I_RESET='0') then
      I_ADR <= "00000000";
    elsif (P_CLK' event and P_CLK='1') then
      if (I_SETA='1'and I_SETA_D='0' and I_MM='1') then
        I_ADR <= P_DATA_SW;
      elsif ((I_INCA='1' or (I_WRITE='1' and I_SSFF='0')) and I_MM='1') then
        I_ADR <= I_ADR + 1;
      elsif (I_DECA='1' and I_MM='1') then
        I_ADR <= I_ADR - 1;
      end if;
    end if;
  end process;

  -- RUN LED
  P_R_LED <= I_SSFF or (P_2_3Hz and I_ERROR);

  -- Start/Stop FF
  P_STOP <= not I_SSFF;
  process(P_CLK, I_RESET)
  begin
    if (I_RESET='0') then
      I_SSFF <= '0';
    elsif (P_CLK' event and P_CLK='1') then
      if (I_STOP='1' or  P_HL='1' or (P_LI='1' and P_MR='1' and P_STEP_SW='1')
          or (P_LI='1' and P_MR='1' and P_BREAK_SW='1' and P_DATA_SW=P_AIN)) the
n
        I_SSFF <= '0';
      elsif (I_RUN='1') then
        I_SSFF <= '1';
      end if;
    end if;
```

```
  end process;

  -- Error FF
  process(P_CLK, I_RESET)
  begin
    if (I_RESET='0') then
      I_ERROR <= '0';
    elsif (P_CLK' event and P_CLK='1') then
      if (P_ER='1') then
        I_ERROR <= '1';
      elsif (I_RUN='1'or I_STOP='1') then
        I_ERROR <= '0';
      end if;
    end if;
  end process;

end RTL;
```