```
--
-- TeC7 VHDL Source Code
--    Tokuyama kousen Educational Computer Ver.7
--
-- Copyright (C) 2002-2012 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/tec_io.vhd : TeC I/O
--

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

entity TEC_IO is
  port ( P_CLK      : in  std_logic;                    -- CLK
         P_2_4kHz   : in  std_logic;                    -- Pi!
         P_75Hz     : in  std_logic;                    -- Key Scan
         P_RESET    : in  std_logic;                    -- Reset
         P_RW       : in  std_logic;
         P_IR       : in  std_logic;
         P_ADDR     : in  std_logic_vector(3 downto 0);
         P_DOUT     : out std_logic_vector(7 downto 0);
         P_DIN      : in  std_logic_vector(7 downto 0);
         P_INT_TXD  : out std_logic;                    -- SIO 送信割り込み
         P_INT_RXD  : out std_logic;                    -- SIO 受信割り込み
         P_INT_TMR  : out std_logic;                    -- TIMER 割り込み
         P_INT_CON  : out std_logic;                    -- コンソール割り込み

         P_INT_SW   : in  std_logic;                    -- コンソール割り込みSW
         P_DATA_SW  : in  std_logic_vector(7 downto 0);
         P_SPK      : out std_logic;
         P_RXD      : in  std_logic;
         P_TXD      : out std_logic;
         P_EXT_IN   : in  std_logic_vector(7 downto 0);
         P_ADC_REF  : out std_logic_vector(7 downto 0);
         P_EXT_OUT  : out std_logic_vector(7 downto 0)
       );
end TEC_IO;

architecture RTL of TEC_IO is
-- Address Decode
signal IOW        : std_logic;
signal IOR        : std_logic;
signal DEC_OUT    : std_logic_vector(11 downto 0);
signal IOR_SIO_DAT: std_logic;
signal IOR_TMR_STA: std_logic;
signal IOW_BUZ    : std_logic;
signal IOW_SPK    : std_logic;
signal IOW_PINT   : std_logic;
signal IOW_SIO_DAT: std_logic;
signal IOW_SIO_CTL: std_logic;
signal IOW_TMR_CNT: std_logic;
signal IOW_TMR_CTL: std_logic;
```

```
signal IOW_EXT_DAT: std_logic;

-- Flip Flop
signal I_SPK      : std_logic;
signal I_BEEP     : std_logic;
signal I_PINT     : std_logic;
signal I_PINT_D   : std_logic;
signal I_PINT_B   : std_logic;

-- SIO
signal Tx_D_Reg   : std_logic_vector(7 downto 0);
signal Tx_S_Reg   : std_logic_vector(7 downto 0);
signal Tx_Out     : std_logic;
signal Tx_Full    : std_logic;
signal Tx_Ena     : std_logic;
signal Tx_Cnt1    : std_logic_vector(7 downto 0);
signal Tx_Cnt2    : std_logic_vector(3 downto 0);
signal Tx_Int_Ena : std_logic;

signal Rx_D_Reg   : std_logic_vector(7 downto 0);
signal Rx_S_Reg   : std_logic_vector(7 downto 0);
signal Rx_Full    : std_logic;
signal Rx_Ena     : std_logic;
signal Rx_Cnt1    : std_logic_vector(7 downto 0);
signal Rx_Cnt2    : std_logic_vector(3 downto 0);
signal Rx_Int_Ena : std_logic;

signal I_RXD1     : std_logic;
signal I_RXD2     : std_logic;

-- TMR
signal TMR_Cnt    : std_logic_vector(7 downto 0); -- タイマーのカウンタ
signal TMR_Max    : std_logic_vector(7 downto 0); -- タイマーの周期
signal TMR_Ena    : std_logic;                    -- タイマーのスタート/ストップ
signal TMR_Int    : std_logic;                    -- タイマー割込み発生中
signal TMR_Int_Ena: std_logic;                    -- タイマー割込み許可

signal I_TMR_Mat  : std_logic;                    -- Max と Cnt が一致した
signal I_INT_TMR_P: std_logic;                    -- 割込み発生時にパルスを発生
signal I_75Hz_D   : std_logic;                    -- 75Hz のエッジ検出用
signal I_75Hz_P   : std_logic;                    -- 75Hz のエッジ検出パルス

-- ADC
signal ADC_CNT    : std_logic_vector(10 downto 0); -- DACへ出力する値
signal ADC_CNT_D  : std_logic_vector(7 downto 0);  -- １クロック前のDAC
signal ADC_TMP0   : std_logic_vector(7 downto 0);  -- CH0用、最大値を探す
signal ADC_TMP1   : std_logic_vector(7 downto 0);  -- CH１用、最大値を探す
signal ADC_TMP2   : std_logic_vector(7 downto 0);  -- CH2用、最大値を探す
signal ADC_TMP3   : std_logic_vector(7 downto 0);  -- CH3用、最大値を探す
signal ADC_VAL0   : std_logic_vector(7 downto 0);  -- CH0用、変換結果
signal ADC_VAL1   : std_logic_vector(7 downto 0);  -- CH1用、変換結果
signal ADC_VAL2   : std_logic_vector(7 downto 0);  -- CH2用、変換結果
signal ADC_VAL3   : std_logic_vector(7 downto 0);  -- CH3用、変換結果
signal ADC_CARRY  : std_logic;                     -- カウンタがオーバフローした
signal ADC_SMP0   : std_logic;                     -- コンパレータ出力を
signal ADC_SMP1   : std_logic;                     --  クロックに同期させた
signal ADC_SMP2   : std_logic;                     --   信号
signal ADC_SMP3   : std_logic;


begin
  -- アドレスデコーダ
  IOW <= P_IR and P_RW;
  IOR <= P_IR and not P_RW;
  with P_ADDR select
    DEC_OUT <=
```

```
  "000000000001" when "0000",
  "000000000010" when "0001",
  "000000000100" when "0010",
  "000000001000" when "0011",
  "000000010000" when "0100",
  "000000100000" when "0101",
  "000001000000" when "0110",
  "000010000000" when "0111",
  "000100000000" when "1000",
  "001000000000" when "1001",
  "010000000000" when "1010",
  "100000000000" when "1011",
  "000000000000" when others;

IOR_SIO_DAT <= IOR and DEC_OUT(2);
IOR_TMR_STA <= IOR and DEC_OUT(5);

IOW_BUZ     <= IOW and DEC_OUT(0);
IOW_SPK     <= IOW and DEC_OUT(1);
IOW_SIO_DAT <= IOW and DEC_OUT(2);
IOW_SIO_CTL <= IOW and DEC_OUT(3);
IOW_TMR_CNT <= IOW and DEC_OUT(4);
IOW_TMR_CTL <= IOW and DEC_OUT(5);
IOW_PINT    <= IOW and DEC_OUT(6);
IOW_EXT_DAT <= IOW and DEC_OUT(7);

-- ブザー,スピーカ,コンソール割り込み部分
P_SPK  <= I_SPK xor ((I_BEEP or I_PINT_B) and P_2_4kHz);
P_INT_CON <= I_PINT and P_INT_SW;
process (P_CLK, P_RESET)
begin
  if (P_RESET='0') then
    I_BEEP   <= '0';
    I_SPK    <= '0';
    I_PINT   <= '0';
    I_PINT_D <= '0';
    I_PINT_B <= '0';
  elsif (P_CLK' event and P_CLK='1') then
    if (IOW_BUZ='1') then
      I_BEEP <= P_DIN(0);
    end if;
    if (IOW_SPK='1') then
      I_SPK  <= P_DIN(0);
    end if;
    if (IOW_PINT='1') then
      I_PINT <= P_DIN(0);
    end if;
    if (I_PINT='1' and P_INT_SW='1') then
      I_PINT_D <= '1';
    end if;
    if (P_75Hz='1' and I_75Hz_D='0' and I_PINT_D='1') then
      I_PINT_B <= '1';
      I_PINT_D <= '0';
    elsif (P_75Hz='0' and I_75Hz_D='1') then
      I_PINT_B <= '0';
    end if;
  end if;
end process;

-- SIO の部分
P_TXD     <= Tx_Out or (not Tx_Ena);
P_INT_TXD <= (not Tx_Full) and Tx_Int_Ena;
P_INT_RXD <=      Rx_Full  and Rx_Int_Ena;

-- Ctl
process (P_CLK, P_RESET)
```

```
begin
  if (P_RESET='0') then
    Tx_Int_Ena <= '0';
    Rx_Int_Ena <= '0';
  elsif (P_CLK'event and P_CLK='1') then
    if (IOW_SIO_CTL='1') then
      Tx_Int_Ena <= P_DIN(7);
      Rx_Int_Ena <= P_DIN(6);
    end if;
  end if;
end process;

-- Tx
process (P_CLK, P_RESET)
begin
  if (P_RESET='0') then
    Tx_D_Reg   <= "00000000";
    Tx_Full    <= '0';
  elsif (P_CLK' event and P_CLK='1') then
    if (IOW_SIO_DAT='1') then
      Tx_D_Reg <= P_DIN;
      Tx_Full  <= '1';
    elsif (Tx_Ena='0') then
      Tx_Full  <= '0';
    end if;
  end if;
end process;

process (P_CLK, P_RESET)
begin
  if (P_RESET='0') then
    Tx_S_Reg <= "00000000";
    Tx_Out <= '0';
    Tx_Ena <= '0';
    Tx_Cnt1 <= "00000000";
    Tx_Cnt2 <= "0000";
  elsif (P_CLK' event and P_CLK='1') then
    if (Tx_Ena='1') then
      if (Tx_Cnt1="11111111") then
        Tx_OUT <= Tx_S_Reg(0);
        Tx_S_Reg(6 downto 0) <= Tx_S_Reg(7 downto 1);
        Tx_S_Reg(7) <= '1';
        Tx_Cnt1 <= "00000000";
        if (Tx_Cnt2="1001") then
          Tx_Ena <= '0';
          Tx_Cnt2 <= "0000";
        else
          Tx_Cnt2 <= Tx_Cnt2 + 1;
        end if;
      else
        Tx_Cnt1 <= Tx_Cnt1 + 1;
      end if;
    elsif (Tx_Full='1') then
      Tx_S_Reg <= Tx_D_Reg;
      Tx_Out <= '0';
      Tx_Ena <= '1';
      Tx_Cnt1 <= "00000000";
    end if;
  end if;
end process;

-- Rx
process (P_CLK, P_RESET)
begin
  if (P_RESET='0') then
    Rx_Full <= '0';
```

```
    I_RXD1 <= '0';
    I_RXD2 <= '0';
  elsif (P_CLK'event and P_CLK='1') then
    I_RXD1 <= P_RXD;
    I_RXD2 <= I_RXD1;
    if (Rx_Cnt1="10000000" and Rx_Cnt2="1001") then
      Rx_Full <= '1';
    elsif (IOR_SIO_DAT='1') then
      Rx_Full <= '0';
    end if;
  end if;
end process;

process (P_CLK, P_RESET)
begin
  if (P_RESET='0') then
    Rx_D_Reg <= "00000000";
    Rx_S_Reg <= "00000000";
    Rx_Ena   <= '0';
    Rx_Cnt1  <= "00000000";
    Rx_Cnt2  <= "0000";
  elsif (P_CLK'event and P_CLK='1') then
    if (Rx_Ena='1') then
      if (Rx_Cnt1="10000000") then
        Rx_S_Reg(6 downto 0) <= Rx_S_Reg(7 downto 1);
        Rx_S_Reg(7)          <= I_RXD2;
        if (Rx_Cnt2="0000" and I_RXD2='1') then
          Rx_Ena             <= '0';
          Rx_Cnt2            <= "0000";
        elsif (Rx_Cnt2="1001") then
          Rx_Ena             <= '0';
          Rx_D_Reg           <= Rx_S_Reg;
          Rx_Cnt2            <= "0000";
        else
          Rx_Cnt2            <= Rx_Cnt2 + 1;
        end if;
      end if;
      Rx_Cnt1 <= Rx_Cnt1 + 1;
    elsif (I_RXD1='0' and I_RXD2='1') then
      Rx_Ena  <= '1';
      Rx_Cnt1 <= "00000000";
    end if;
  end if;
end process;

-- Timer
I_TMR_Mat <= '1' when (TMR_CNT=TMR_Max) else '0';  -- カウンタ = 目的の値
I_75Hz_P <= not P_75Hz and I_75Hz_D and TMR_Ena;   -- 立下がりを検出
I_INT_TMR_P <= I_75Hz_P and I_TMR_Mat;             -- 割込み発生パルス
P_INT_TMR <= I_INT_TMR_P and TMR_Int_Ena;
process (P_CLK, P_RESET)
begin
  if (P_RESET='0') then
    TMR_CNT <= "00000000";
    TMR_Max <= "01001010";
    TMR_Ena <= '0';
    TMR_Int <= '0';
    TMR_Int_Ena <= '0';
    I_75Hz_D <= '0';
  elsif (P_CLK'event and P_CLK='1') then
    I_75Hz_D <= P_75Hz;

    -- タイマーの周期レジスタ
    if (IOW_TMR_CNT='1') then
      TMR_Max <= P_DIN;             -- 周期を変更
    end if;
```

```
    -- タイマーのスタートストップ制御
    if (IOW_TMR_CNT='1') then
      TMR_Ena <= '0';               -- 変更時に、タイマーは自動的に止まる
    elsif (IOW_TMR_CTL='1') then
      TMR_Ena <= P_DIN(0);          -- タイマーのスタートストップ
      TMR_Int_Ena <= P_DIN(7);      -- タイマーの割込み許可
    end if;

    -- タイマーのカウンタ制御
    if (IOW_TMR_CTL='1' or I_INT_TMR_P='1') then
      TMR_CNT <= "00000000";        -- Start/Stop、コンペアマッチでリセット
    elsif (I_75Hz_P='1') then
      TMR_CNT <= TMR_CNT + 1;       -- それ以外ではカウントアップ
    end if;

    -- ポーリングビットの制御
    if (IOW_TMR_CTL='1') then
      TMR_Int <= '0';               -- CPU がタイマーをスタートしたらリセット
    elsif (I_INT_TMR_P='1') then
      TMR_Int <= '1';               -- 割り込み発生のと同時にセット
    end if;
  end if;
end process;

-- ADC
P_ADC_REF <= ADC_CNT(10 downto 3);
process (P_CLK, P_RESET)
begin
  if (P_CLK'event and P_CLK='1') then
    ADC_CNT   <= ADC_CNT + 1;
  end if;
  if (P_CLK'event and P_CLK='1' and ADC_CNT(2 downto 0)="111") then
    ADC_CNT_D <= ADC_CNT(10 downto 3);
    if (ADC_CNT_D="11111111") then
      ADC_CARRY <= '1';
    else
      ADC_CARRY <= '0';
    end if;

    ADC_SMP0  <= P_EXT_IN(0);
    ADC_SMP1  <= P_EXT_IN(1);
    ADC_SMP2  <= P_EXT_IN(2);
    ADC_SMP3  <= P_EXT_IN(3);

    if (ADC_SMP0='1') then
      ADC_TMP0 <= ADC_CNT_D;
    end if;
    if (ADC_SMP1='1') then
      ADC_TMP1 <= ADC_CNT_D;
    end if;
    if (ADC_SMP2='1') then
      ADC_TMP2 <= ADC_CNT_D;
    end if;
    if (ADC_SMP3='1') then
      ADC_TMP3 <= ADC_CNT_D;
    end if;

    if (ADC_CARRY='1') then
      ADC_VAL0 <= ADC_TMP0;
      ADC_VAL1 <= ADC_TMP1;
      ADC_VAL2 <= ADC_TMP2;
      ADC_VAL3 <= ADC_TMP3;
    end if;
  end if;
end process;
```

```
  -- I/O ポート
  process (P_CLK, P_RESET)
  begin
    if (P_RESET='0') then
      P_EXT_OUT <= "00000000";
    elsif (P_CLK'event and P_CLK='1') then
      if (IOW_EXT_DAT='1') then
        P_EXT_OUT <= P_DIN;
      end if;
    end if;
  end process;

  -- Data Bus の出力選択
  P_DOUT <= P_DATA_SW when (P_ADDR="0000") else
            P_DATA_SW when (P_ADDR="0001") else
            Rx_D_Reg  when (P_ADDR="0010") else
            (not Tx_Full & Rx_Full & "000000") when (P_ADDR="0011") else
            TMR_CNT   when (P_ADDR="0100") else
            (TMR_Int & "0000000") when (P_ADDR="0101") else
            (P_EXT_IN(7 downto 4) & ADC_VAL3(7) & ADC_VAL2(7)
              & ADC_VAL1(7) & ADC_VAL0(7)) when (P_ADDR="0111") else
  --        P_EXT_IN when (P_ADDR="0111") else
            ADC_VAL0 when (P_ADDR="1000") else
            ADC_VAL1 when (P_ADDR="1001") else
            ADC_VAL2 when (P_ADDR="1010") else
            ADC_VAL3 when (P_ADDR="1011") else
            "00000000";
end RTL;
```