CH9_KEYboard_1.vhd

```
--4x4 鍵盤_基本測試:按下放開再處理
--107.01.01 版
--EP3C16Q240C8 50MHz LEs:15,408 PINs:161 ,gckp31 ,rstP99

Library IEEE;                          --連結零件庫
Use IEEE.std_logic_1164.all;           --引用套件
Use IEEE.std_logic_unsigned.all;       --引用套件

entity CH9_KEYboard_1 is
port(gckp31,rstP99:in std_logic;    --系統頻率,系統 reset
     keyi:in std_logic_vector(3 downto 0);      --鍵盤輸入
     keyo:buffer std_logic_vector(3 downto 0); --鍵盤輸出
     LED1_16:buffer std_logic_vector(15 downto 0);--LED 顯示
     --4 位數掃描式顯示器
     SCANo:buffer std_logic_vector(3 downto 0);--掃瞄器輸出
     Disp7S:buffer std_logic_vector(7 downto 0)--計數位數解碼輸出
    );
end entity CH9_KEYboard_1;

architecture Albert of CH9_KEYboard_1 is
    signal FD:std_logic_vector(24 downto 0);--除頻器
    signal kn,kin:integer range 0 to 15;    --新鍵值,原鍵值
    signal kok:std_logic_vector(3 downto 0):="0000";--鍵盤偵測狀態
    signal i:integer range 0 to 3;          --鍵盤輸入偵測指標

begin

--LED_on_off:取用頻率高於鍵盤操作頻率--
LED_on_off:process(FD(12))
variable sw:std_logic;
begin
    if rstP99='0' then          --系統 reset
        LED1_16<=(others=>'0');--取消所有燈號
        kin<=0;                --設原鍵值 0
        sw:='0';               --可接收新鍵值
    elsif (rising_edge(FD(12))) then
        if (kok=10) and sw='0' then     --鍵盤已完成狀態
            kin<=kn;                    --取得新鍵值
            LED1_16(kn)<=not LED1_16(kn);--新鍵值燈號
            sw:='1';                    --停止接收新鍵值
        elsif kok=1 then                --鍵盤已重啟
            sw:='0';                    --可再接收新鍵值
        end if;
    end if;
end process LED_on_off;

--keyboard:鍵盤操作頻率低於取用者頻率--
keyboard:process (FD(13))
begin
    if kok=0 or rstP99='0' then     --重置鍵盤,系統 reset
```

```
        keyo<="1110";                   --準備鍵盤輸出信號
        kok<="0001";                    --預設鍵盤未完成、無按鍵狀態
        kn<=0;                          --鍵值由 0 開始
        i<=0;                           --鍵盤偵測指標由 0 開始
    elsif (rising_edge(FD(13))) then
        if (kok/=1) then                --有按鍵狀態

            --適當調整 kok 值可使鍵盤運作順暢
            if keyi=15 then             --判斷按鍵全放開
                if kok<5 then           --初期:鍵盤防彈跳過程狀態
                    kok<=kok-1;         --如是雜訊將重啟鍵盤
                else
                    kok<=kok+1;         --鍵盤防彈跳過程狀態(後期)
                end if;
            else
                if kok>9 then           --取用了:鍵盤防彈跳過程狀態
                    kok<="1011";        --(後期)
                elsif kok>4 then        --尚未取用:鍵盤防彈跳過程狀態
                    kok<="0101";
                else
                    kok<=kok+1;         --初期:鍵盤防彈跳過程狀態
                end if;
            end if;

        elsif keyi(i)='0' then          --偵測按鍵按下狀態
            kok<="0010";                --設有按鍵狀態
            keyo<="0000";               --設偵測所有按鍵
        else                            --無按鍵按下狀態
            kn<=kn+1;                   --調整鍵值
            keyo<=keyo(2 downto 0) & keyo(3);--調整鍵盤輸出
            if keyo(3)='0' then         --是否要調整鍵盤偵測指標
                i<=i+1;                 --調整鍵盤偵測指標
            end if;
        end if;
    end if;
end process keyboard;

SCANo<="1110";
--七段顯示器解碼 0123456789AbCdEF--pgfedcba
with kin select
Disp7S <=   "11000000" when 0,  --0
            "11111001" when 1,  --1
            "10100100" when 2,  --2
            "10110000" when 3,  --3
            "10011001" when 4,  --4
            "10010010" when 5,  --5
            "10000010" when 6,  --6
            "11111000" when 7,  --7
            "10000000" when 8,  --8
            "10010000" when 9,  --9
            "10001000" when 10, --A
            "10000011" when 11, --b
            "11000110" when 12, --C
```

```
            "10100001" when 13, --d
            "10000110" when 14, --E
            "10001110" when 15; --F

--除頻器--
Freq_Div:process(gckP31)            --系統頻率 gckP31:50MHz
begin
    if rstP99='0' then              --系統重置
        FD<=(others=>'0');          --除頻器:歸零
    elsif rising_edge(gckP31) then --50MHz
        FD<=FD+1;                   --除頻器:2 進制上數(+1)計數器
    end if;
end process Freq_Div;

end Albert;
```

CH9_KEYboard_2.vhd

```
--4x4 鍵盤_基本測試:按下馬上處理
--107.01.01 版
--EP3C16Q240C8 50MHz LEs:15,408 PINs:161 ,gckp31 ,rstP99

Library IEEE;                       --連結零件庫
Use IEEE.std_logic_1164.all;        --引用套件
Use IEEE.std_logic_unsigned.all;    --引用套件

entity CH9_KEYboard_2 is
port(gckP31,rstP99:in std_logic;    --系統頻率,系統 reset
     keyi:in std_logic_vector(3 downto 0);      --鍵盤輸入
     keyo:buffer std_logic_vector(3 downto 0); --鍵盤輸出
     LED1_16:buffer std_logic_vector(15 downto 0);--LED 顯示
     --4 位數掃描式顯示器
     SCANo:buffer std_logic_vector(3 downto 0);--掃瞄器輸出
     Disp7S:buffer std_logic_vector(7 downto 0)--計數位數解碼輸出
    );
end entity CH9_KEYboard_2;

architecture Albert of CH9_KEYboard_2 is
    signal FD:std_logic_vector(24 downto 0);--除頻器
    signal kn,kin:integer range 0 to 15;   --新鍵值,原鍵值
    signal kok:std_logic_vector(3 downto 0):="0000";--鍵盤偵測狀態
    signal i:integer range 0 to 3;          --鍵盤輸入偵測指標
    type Disp7DataT is array(0 to 3) of integer range 0 to 15;
    --顯示表格式
    signal Disp7Data:Disp7DataT;            --顯示表
    signal scanP:integer range 0 to 3;      --掃瞄器指標
    signal times:integer range 0 to 1023;  --計時器

begin
```

```
--LED_on_off:取用者頻率高於鍵盤操作頻率--
LED_on_off_7s:process (FD(12))
variable sw:std_logic;
begin
    if rstP99='0' then              --系統 reset
        LED1_16<=(others=>'0');     --取消所有燈號
        kin<=0;                     --設原鍵值 0
        sw:='0';                    --可接收新鍵值
        Disp7Data<=(0,0,0,0);
        times<=1023;
    elsif (rising_edge(FD(12))) then
        if (kok=5) and sw='0' then --鍵盤已完成狀態
            kin<=kn;                --取得新鍵值
            LED1_16(kn)<=not LED1_16(kn);--新鍵值燈號
            if kn<10 then           --0~9 載入顯示表
                Disp7Data(0)<=kn;   --由個位數推入
                for i in 0 to 2 loop--其餘往高位進一位
                    Disp7Data(i+1)<=Disp7Data(i);
                end loop;
            else                    --命令
                case kn is
                    when 10=>       --歸零
                        Disp7Data<=(0,0,0,0);
                    when 11=>       --後退一位
                        for i in 0 to 2 loop
                            Disp7Data(i)<=Disp7Data(i+1);
                        end loop;
                        Disp7Data(3)<=0;
                    when others=>   --其餘紀錄於 kin
                        null;
                end case;
            end if;
            sw:='1';                --停止接收新鍵值
            times<=1023;            --計時重設
        else
            if kok=1 then           --鍵盤已重啟
                sw:='0';            --可再接收新鍵值
            end if;
            times<=times-1;         --命令執行
            if times=0 then         --可執行命令
                times<=1023;        --計時重設
                case kin is
                    when 12=>       -- +1
                        if  (Disp7Data(0)+Disp7Data(1)+
                             Disp7Data(2)+Disp7Data(3))/=0 then
                            if Disp7Data(0)/=9  then
                                Disp7Data(0)<=Disp7Data(0)+1;
                            else Disp7Data(0)<=0;
                            if Disp7Data(1)/=9 then
                                Disp7Data(1)<=Disp7Data(1)+1;
                            else Disp7Data(1)<=0;
                            if Disp7Data(2)/=9 then
```

```
                                    Disp7Data(2)<=Disp7Data(2)+1;
                                else Disp7Data(2)<=0;
                                if Disp7Data(3)/=9 then
                                    Disp7Data(3)<=Disp7Data(3)+1;
                                else Disp7Data(3)<=0;
                                end if; end if; end if; end if;
                            end if;
                        when 13=>       -- -1
                            if (Disp7Data(0)+Disp7Data(1)+
                                Disp7Data(2)+Disp7Data(3))/=0 then
                                if Disp7Data(0)/=0 then
                                    Disp7Data(0)<=Disp7Data(0)-1;
                                else Disp7Data(0)<=9;
                                if Disp7Data(1)/=0 then
                                    Disp7Data(1)<=Disp7Data(1)-1;
                                else Disp7Data(1)<=9;
                                if Disp7Data(2)/=0 then
                                    Disp7Data(2)<=Disp7Data(2)-1;
                                else Disp7Data(2)<=9;
                                    Disp7Data(3)<=Disp7Data(3)-1;
                                end if;end if;end if;
                            end if;
                        when 14=>       --右旋
                            Disp7Data(3)<=Disp7Data(0);
                            for i in 0 to 2 loop
                                Disp7Data(i)<=Disp7Data(i+1);
                            end loop;
                        when 15=>       --左旋
                            Disp7Data(0)<=Disp7Data(3);
                            for i in 0 to 2 loop
                                Disp7Data(i+1)<=Disp7Data(i);
                            end loop;
                        when others=>   --0~11
                            null;
                    end case;
                end if;
            end if;
        end if;
    end process LED_on_off_7s;

    --keyboard:鍵盤操作頻率低於取用者頻率--
    keyboard:process (FD(13))
    begin
        if kok=0 or rstP99='0' then--重置鍵盤,系統 reset
            keyo<="1110";           --準備鍵盤輸出信號
            kok<="0001";            --預設鍵盤未完成、無按鍵狀態
            kn<=0;                  --鍵值由 0 開始
            i<=0;                   --鍵盤偵測指標由 0 開始
        elsif (rising_edge(FD(13))) then
            if (kok/=1) then        --有按鍵狀態

                --適當調整 kok 值可使鍵盤運作順暢
                if keyi=15 then     --判斷按鍵全放開
```

```
                if kok<5 then   --初期:鍵盤防彈跳過程狀態
                    kok<=kok-1; --如是雜訊將重啟鍵盤
                else
                    kok<=kok+1; --鍵盤防彈跳過程狀態(後期)
                end if;
            else
                if kok>4 then   --取用了:鍵盤防彈跳過程狀態
                    kok<="0110";
                else
                    kok<=kok+1; --初期:鍵盤防彈跳過程狀態
                end if;
            end if;

        elsif keyi(i)='0' then --偵測按鍵按下狀態
            kok<="0010";       --設有按鍵狀態
            keyo<="0000";      --設偵測所有按鍵
        else                   --無按鍵按下狀態
            kn<=kn+1;          --調整鍵值
            keyo<=keyo(2 downto 0) & keyo(3);--調整鍵盤輸出
            if keyo(3)='0' then--是否要調整鍵盤偵測指標
                i<=i+1;        --調整鍵盤偵測指標
            end if;
        end if;
    end if;
end process keyboard;

--4位數掃瞄器--
scan_P:process(FD(17),rstP99)
begin
    if rstP99='0' then
        scanP<=0;           --位數取值指標
        SCANo<="1111";      --掃瞄信號 all off
    elsif rising_edge(FD(17)) then
        scanP<=scanP+1;     --位數取值指標遞增
        SCANo<=SCANo(2 downto 0)&SCANo(3);
        if scanP=3 then     --最後一位數了
            scanP<=0;       --位數取值指標重設
            SCANo<="1110";  --掃瞄信號重設
        end if;
    end if;
end process scan_P;

--七段顯示器解碼 0123456789AbCdEF--pgfedcba
with Disp7Data(scanP) select
Disp7S <=   "11000000" when 0,  --0
            "11111001" when 1,  --1
            "10100100" when 2,  --2
            "10110000" when 3,  --3
            "10011001" when 4,  --4
            "10010010" when 5,  --5
            "10000010" when 6,  --6
            "11111000" when 7,  --7
            "10000000" when 8,  --8
```

```
            "10010000" when 9,  --9
            "10001000" when 10, --A
            "10000011" when 11, --b
            "11000110" when 12, --C
            "10100001" when 13, --d
            "10000110" when 14, --E
            "10001110" when 15; --F

--除頻器--
Freq_Div:process(gckP31)            --系統頻率 gckP31:50MHz
begin
    if rstP99='0' then              --系統重置
        FD<=(others=>'0');          --除頻器:歸零
    elsif rising_edge(gckP31) then --50MHz
        FD<=FD+1;                   --除頻器:2 進制上數(+1)計數器
    end if;
end process Freq_Div;

end Albert;
```

CH9_ROTATE_ENCODER_3.vhd

```
--旋轉編碼器_基本測試
--EP3C16Q240C8 50MHz LEs:15,408 PINs:161 ,gckp31 ,rstP99

Library IEEE;                           --連結零件庫
Use IEEE.std_logic_1164.all;            --引用套件
Use IEEE.std_logic_unsigned.all;        --引用套件

entity CH9_ROTATE_ENCODER_3 is
port(gckp31,rstP99:in std_logic;        --系統頻率,系統 reset
     APi,BPi,PBi:in std_logic;          --旋轉編碼器
     LED1_16:buffer std_logic_vector(15 downto 0)--LED 顯示
    );
end entity CH9_ROTATE_ENCODER_3;

architecture Albert of CH9_ROTATE_ENCODER_3 is
    signal FD: std_logic_vector(25 downto 0);      --除頻器
    signal APic,BPic,PBic:std_logic_vector(2 downto 0):="000";
    --防彈跳計數器
    signal mode1,mode2:integer range 0 to 3;
    --樣板,操作模式
begin

--旋轉編碼器介面電路--
EncoderInterface:process(APi,PBi,rstP99)
begin
    if rstP99='0' then
        mode2<=0;                           --由 0 開始
```

```
    elsif rising_edge(PBic(2)) then         --偵測到 UD 信號的升緣時
        mode2<=mode2+1;                     --下一次變更依據
    end if;

    if rstP99='0' or PBic(2)='0' then
        mode1<=mode2;                       --依 mode2 變更選項
        if mode1=0 then                     --依 mode2 載入樣板
            LED1_16<=(others=>'0');
        elsif mode1=1 then
            LED1_16<="1100110011001100";
        elsif mode1=2 then
            LED1_16<="1111000000001111";
        else
            LED1_16<="1010101010101010";
        end if;
    elsif rising_edge(APic(2)) then         --偵測到 UD 信號的升緣時
        case mode1 is
            when 0=>
                if BPi='1' then             --左旋
                    LED1_16<=not LED1_16(0)& LED1_16(15 downto 1);
                else                        --右旋
                    LED1_16<=LED1_16(14 downto 0) & not LED1_16(15);
                end if;
            when 1=>
                if BPi='1' then             --左旋
                    LED1_16<=LED1_16(0)& LED1_16(15 downto 1);
                else                        --右旋
                    LED1_16<=LED1_16(14 downto 0) & LED1_16(15);
                end if;
            when 2=>
                if BPi='1' then             --外向內
                    LED1_16<=LED1_16(14 downto 8) & LED1_16(15)&
                    LED1_16(0)& LED1_16(7 downto 1);
                else                        --內向外
                    LED1_16<=LED1_16(8)& LED1_16(15 downto 9)&
                    LED1_16(6 downto 0) & LED1_16(7);
                end if;
            when 3=>
                if BPi='1' then             --左反相
                    LED1_16<=not LED1_16(15 downto 8)&LED1_16(7 downto
0);
                else                        --右反相
                    LED1_16<=LED1_16(15 downto 8)&not LED1_16(7 downto
0);
                end if;
        end case;
    end if;
end process EncoderInterface;

-- 防彈跳電路
Debounce:process(FD(8))             --旋轉編碼器防彈跳頻率
begin

    --APi 防彈跳與雜訊
```

```
    if APi=APic(2) then          --若 APi 等於 APic 最左邊位元
        APic<=APic(2) & "00";    --則 APi 等於 APic(2)右邊位元歸零
    elsif rising_edge(FD(8)) then  --取樣頻率約為 97.7KHz
        APic<=APic+1;            --否則隨 F1 的升緣，APic 計數器遞增
    end if;

    --BPi 防彈跳與雜訊
    if BPi=BPic(2) then          --若 BPi 等於 BPic 最左邊位元
        BPic<=BPic(2)& "00";     --則 BPi 等於 BPic(2)右邊位元歸零
    elsif rising_edge(FD(8)) then  --取樣頻率約為 97.7KHz
        BPic<=BPic+1;            --否則隨 F1 的升緣，BPic 計數器遞增
    end if;

    --PBi 防彈跳與雜訊
    if PBi=PBic(2) then          --若 PBi 等於 PBic 最左邊位元
        PBic<=PBic(2)& "00";     --則 PBic(2)右邊位元歸零
    elsif rising_edge(FD(16)) then --取樣頻率約為 381Hz
        PBic<=PBic+1;            --否則隨 F1 的升緣，PBic 計數器遞增
    end if;
end process Debounce;

--除頻器--
Freq_Div:process(gckP31)             --系統頻率 gckP31:50MHz
begin
    if rstP99='0' then               --系統重置
        FD<=(others=>'0');           --除頻器:歸零
    elsif rising_edge(gckP31) then --50MHz
        FD<=FD+1;                    --除頻器:2 進制上數(+1)計數器
    end if;
end process Freq_Div;

end Albert;
```