開關彈跳VHDL—0~99顯示

```
library ieee ;
use ieee.std_logic_1164.all ;
use ieee.std_logic_unsigned.all ;
use ieee.std_logic_arith.all ;
--******************************
entity bounce is
 port ( clk2: in std_logic;
        sw1: in std_logic;
        scan: out std_logic_vector(3 downto 0);
        seg : out std_logic_vector(0 to 6)) ;
end bounce;
--******************************
architecture A_bounce of bounce is
signal c1,c2,bcd    :   integer range 0 to 9;
signal clk_in : std_logic;

begin
process(clk2)
begin
    wait until clk2= '1';
    if sw1='1' then
        clk_in<='1';
    else clk_in<='0';
    end if;
end process;
--*************************
inc_1:
process(clk_in)
    begin
     wait until clk_in='1' ;
    if c1<9 then c1<=c1+1;
        elsif c2<9 then
            c1<=0;
            c2<=c2+1;
        else c1<=0;
             c2<=0;
         end if;
end process inc_1;
--***********************************
Scan_a:
process(Clk2)
 variable Scan1 : integer range 0 to 1;
  begin
  --if Clk2='1' and Clk2'event then
```

產生加 1 之觸發訊號，無防止開關彈跳設計，SW1 開關動作 1 次，可能產生 N 次的 clk_in 變化，視 clk2 系統同步訊號的時間及開關本身的結構而會產生不同的觸發訊號 clk_in，所以 clk2 的觸發訊號時間越長，彈跳的造成的錯誤就會降低，約 100HZ 以下時就可消除彈跳的現象

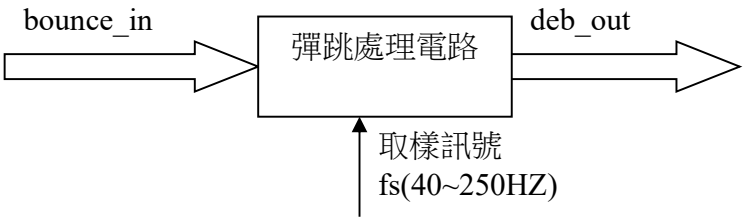
由 0 到 99 的 加 1 訊號

```
    wait until clk2='1' ;
    if (Scan1=0) then
         Scan <= "1101" ;
         bcd <= c2;
    else Scan <= "1110" ;
         bcd <= c1;
    end if ;
    if Scan1 = 1 then Scan1 := 0 ;
     else Scan1 := Scan1 + 1 ;
    end if ;
end process scan_a ;
--********************************
with bcd select
 seg <= "1111110"  when 0 ,
        "0110000"  when 1 ,
        "1101101"  when 2 ,
        "1111001"  when 3 ,
        "0110011"  when 4 ,
        "1011011"  when 5 ,
        "0011111"  when 6 ,
        "1110000"  when 7 ,
        "1111111"  when 8 ,
        "1110011"  when 9 ,
        "1001111"  when others ;
  end A_bounce  ;
```

兩位數之七段顯示器掃描訊號

開關之防彈跳

按鍵彈跳雜訊一般發生在高低電位轉換的4ms內，如果按鍵的機構設計不佳，有可能會產生大於4ms的彈跳雜訊，彈跳消除電路的設計需要一個取樣頻率，使彈跳雜訊最多被取樣一次，而輸入訊號穩態至少需被連續取樣兩次，這樣就表示輸入訊號已經穩定。(取樣頻率範圍約為 40HZ < fs < 250HZ)



例用 Mealy 狀態機 設計之防彈跳開關VHDL

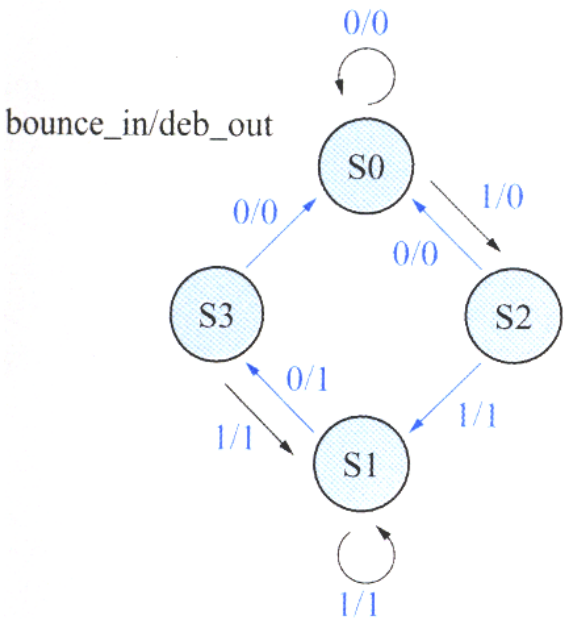
```
LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;
USE IEEE.std_logic_unsigned.all;
USE IEEE.std_logic_arith.all;
ENTITY debounce IS
PORT(
      fs:IN STD_LOGIC;
      bounce_in:in std_logic;
      deb_out:out STD_LOGIC
     );
END debounce;
ARCHITECTURE beh OF debounce IS
type state is (s0,s1,s2,s3);
begin
debounce:process(fs)
  variable  pre_state: state;
begin
  if fs='1' and fs'event then
     case pre_state is
      when s0 =>
      if bounce_in= '0' then
         pre_state := s0;
         deb_out<= '0';
      else
         pre_state := s2;
         deb_out<= '0';
      end if;

      when s1 =>
      if bounce_in= '1' then
         pre_state := s1;
         deb_out<= '1';
      else
         pre_state := s3;
         deb_out<= '1';
      end if;

      when s2 =>
      if bounce_in= '1' then
         pre_state := s1;
         deb_out<= '1';
      else
         pre_state := s0;
         deb_out<= '0';
      end if;
```

type 語法
type <型別名稱> is <型別內容>
例如：
type bit is ('0','1')

狀態 0

狀態 1

狀態 2



```
      when s3 =>
      if bounce_in= '1' then
         pre_state := s1;
         deb_out<= '1';
      else
         pre_state := s0;
         deb_out<= '0';
      end if;

      when others =>
      null;

     end case;
  end if;
 end process debounce;
end beh;
```

狀態 3

防彈跳開關VHDL(1)--0~99顯示

```
library ieee ;
use ieee.std_logic_1164.all ;
use ieee.std_logic_unsigned.all ;
use ieee.std_logic_arith.all ;
--******************************
entity debounce_99 is
 port ( clk2: in std_logic;
        sw1: in std_logic;
        scan: out std_logic_vector(3 downto 0);
        seg : out std_logic_vector(0 to 6)) ;
end debounce_99;
--******************************
architecture A_bounce of debounce_99 is
signal c1,c2,bcd    :   integer range 0 to 9;
signal clk_in : std_logic;
--*****************************************
component debounce
PORT(
     fs:IN STD_LOGIC;
     bounce_in:in std_logic;
     deb_out:out STD_LOGIC
    );
end component;
--******************************
begin
deb_clk_in:debounce
    port map(clk2,sw1,clk_in);
--**************************
inc_1:
process(clk_in)
    begin
     wait until clk_in='1' ;
    if c1<9 then c1<=c1+1;
        elsif c2<9 then
            c1<=0;
            c2<=c2+1;
        else c1<=0;
             c2<=0;
         end if;
end process inc_1;
--**********************************
Scan_a:
process(Clk2)
 variable Scan1 : integer range 0 to 1;
  begin
```

呼叫防彈跳程式 debounce

```
    wait until clk2='1' ;
    if (Scan1=0) then
         Scan <= "1101" ;
         bcd <= c2;
    else Scan <= "1110" ;
         bcd <= c1;
    end if ;
    if Scan1 = 1 then Scan1 := 0 ;
     else Scan1 := Scan1 + 1 ;
    end if ;
end process scan_a ;
--********************************
with bcd select
 seg <= "1111110"  when 0 ,
        "0110000"  when 1 ,
        "1101101"  when 2 ,
        "1111001"  when 3 ,
        "0110011"  when 4 ,
        "1011011"  when 5 ,
        "0011111"  when 6 ,
        "1110000"  when 7 ,
        "1111111"  when 8 ,
        "1110011"  when 9 ,
        "1001111"  when others ;
  end A_bounce  ;
```

防彈跳開關VHDL(2)--0~99顯示(防彈跳開關訊號除頻至約150HZ)

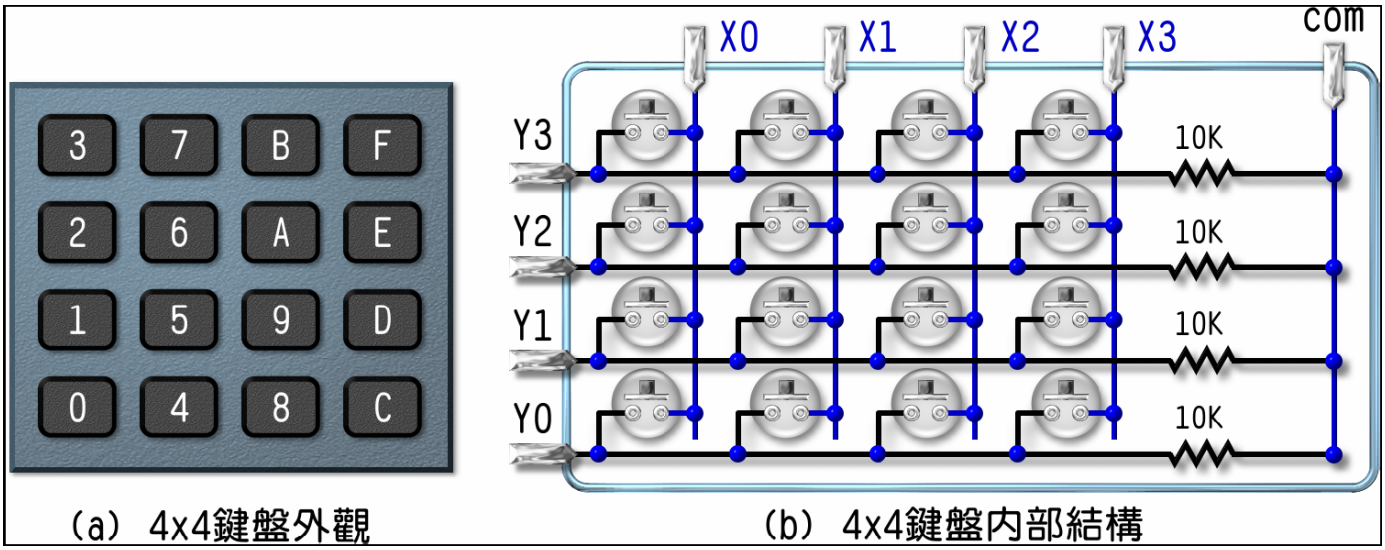
```
library ieee ;
use ieee.std_logic_1164.all ;
use ieee.std_logic_unsigned.all ;
use ieee.std_logic_arith.all ;
--******************************
entity debounce_99 is
 port ( clk2: in std_logic;
        sw1: in std_logic;
        scan: out std_logic_vector(3 downto 0);
        seg : out std_logic_vector(0 to 6)) ;
end debounce_99;
--******************************
architecture A_bounce of debounce_99 is
signal c1,c2,bcd    :   integer range 0 to 9;
signal clk_in,fout : std_logic;

--*****************************************
component debounce
PORT(
     fs:IN STD_LOGIC;
     bounce_in:in std_logic;
     deb_out:out STD_LOGIC
    );
end component;
--****************************
begin
mod_6:process(clk2)
variable cnt:integer range 0 to 3;
  begin
   if clk2='1' and clk2'event then
     if cnt=0 then
        fout<=not fout;
        cnt:=3;
     else
        cnt:=cnt-1;
     end if;
    end if;
  end process;
--*******************************
deb_clk_in:debounce
    port map(fout,sw1,clk_in);
--*************************
inc_1:
process(clk_in)
    begin
```

clk2 七節顯示器掃描顯示觸發訊號，再除 6 成為開關之防彈跳觸發訊號。

```
     wait until clk_in='1' ;
    if c1<9 then c1<=c1+1;
        elsif c2<9 then
            c1<=0;
            c2<=c2+1;
        else c1<=0;
             c2<=0;
         end if;
end process inc_1;
--**********************************
Scan_a:
process(Clk2)
 variable Scan1 : integer range 0 to 1;
  begin
    wait until clk2='1' ;
    if (Scan1=0) then
         Scan <= "1101" ;
         bcd <= c2;
    else Scan <= "1110" ;
         bcd <= c1;
    end if ;
    if Scan1 = 1 then Scan1 := 0 ;
     else Scan1 := Scan1 + 1 ;
    end if ;
end process scan_a ;
--*******************************
with bcd select
 seg <= "1111110"  when 0 ,
        "0110000"  when 1 ,
        "1101101"  when 2 ,
        "1111001"  when 3 ,
        "0110011"  when 4 ,
        "1011011"  when 5 ,
        "0011111"  when 6 ,
        "1110000"  when 7 ,
        "1111111"  when 8 ,
        "1110011"  when 9 ,
        "1001111"  when others ;
  end A_bounce  ;
```

◆鍵盤掃描原理：

如圖之(a)所示為4x4 鍵盤，而(b)為其內部結構，其中包含4行(column)、4列(row)，構成一個4x4 的陣列，在此將每行連接端點定名為X0、X1、X2 及X3，而每列連接端點定名為Y0、Y1、Y2 及Y3。另外，每列各連接一個電阻器到共同接點(com)上。依掃瞄方式的不同，com 可能連接到VCC 或GND，當我們要進行鍵盤掃瞄時，則將掃瞄信號送至 **X0** 到 **X3**，再由 **Y0** 至 **Y3** 讀取鍵盤狀態，即可判斷哪個按鍵被按下。鍵盤掃瞄的方式有兩種，即低態掃瞄與高態掃瞄，如下說明：



1. 低態掃瞄

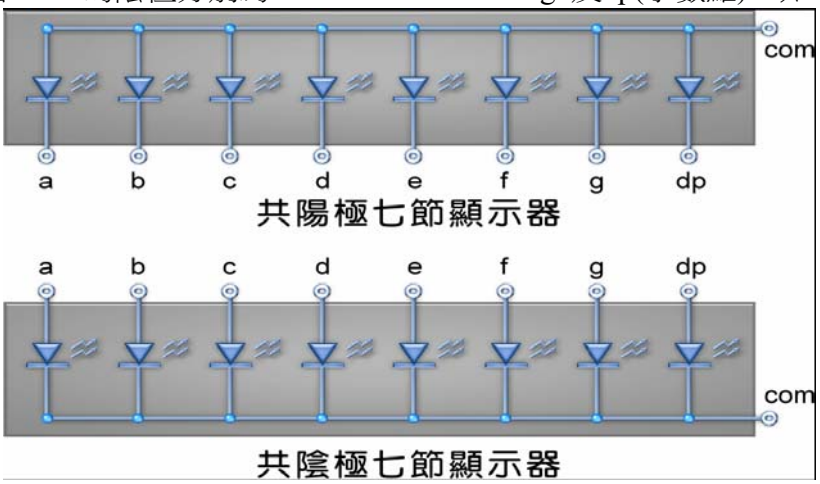
低態掃瞄是將共同點com 連接VCC，沒有按何按鍵被按下時，Y3、Y2、Y1、Y0 端點能保持為高態(即1)。送入X3、X2、X1、X0 的掃瞄信號之中，只有一個為低態(即0)，其餘三個為高態。通常以低態掃瞄較常被使用。

2. 高態掃瞄

高態掃瞄是將共同點com 接地(GND)，沒有按何按鍵被按下時，Y3、Y2、Y1、Y0 端點能保持為低態(即0)。送入X3、X2、X1、X0 的掃瞄信號之中，只有一個為高態(即1)，其餘三個為低態。

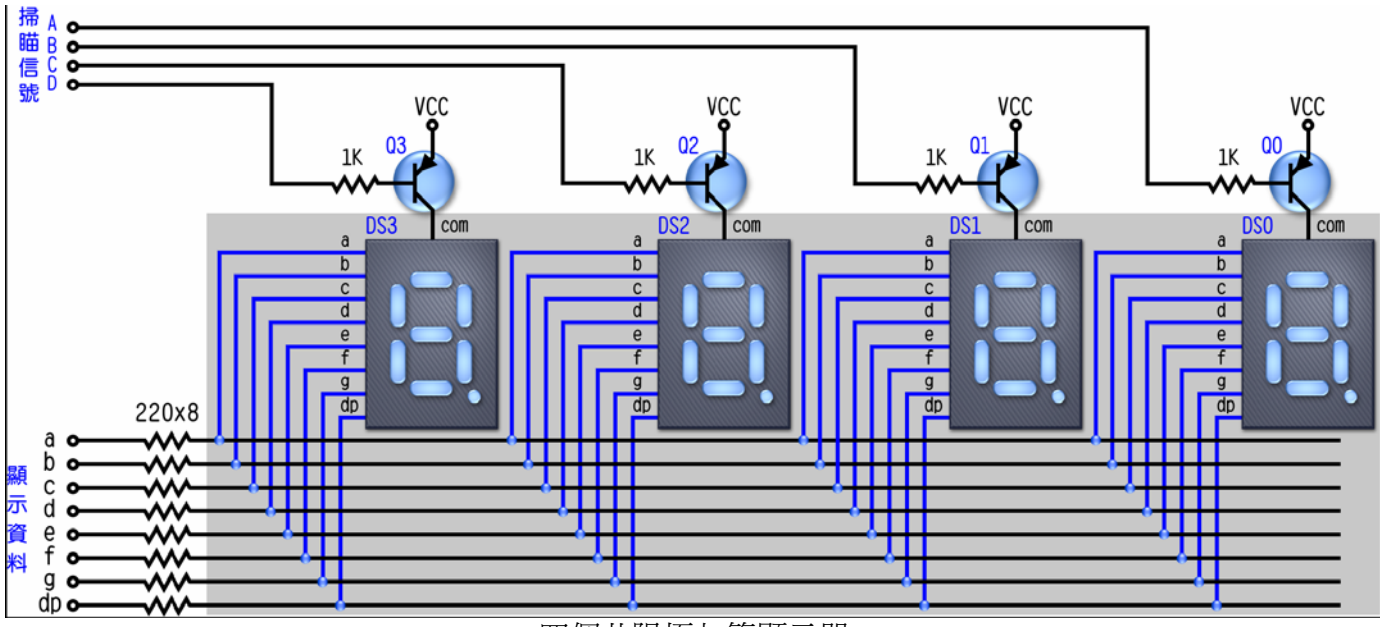
◆七節顯示器：

基本上，七節顯示器可分為共陽極與共陰極兩種，共陽極就是把所有LED 的陽極連接到共同接點com，而每個LED 的陰極分別為a、b、c、d、e、f、g 及dp(小數點)，如下圖所示：



同樣地，共陰極就是把所有LED 的陰極連接到共同接點com，而每個LED 的陽極分別為a、b、c、d、e、f、g 及dp(小數點)。

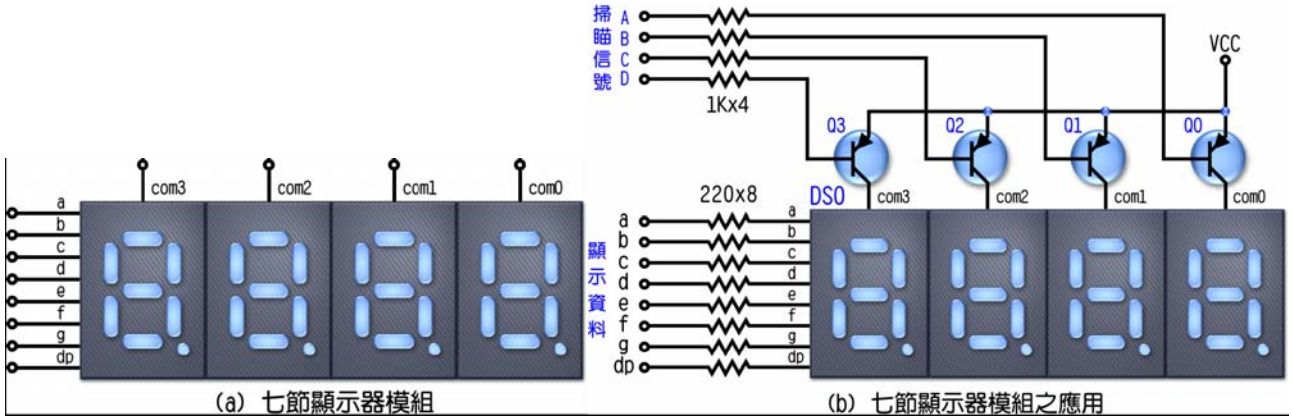
同時使用多個七節顯示器時，可採用掃瞄式顯示，也就是將每個七節顯示器的a、b…g 都連接在一起，再使用電晶體分別驅動每個七節顯示器的共同接腳com。以四個共陽極七節顯示器為例，如下圖所示：



四個共陽極七節顯示器

其顯示方式是將第一個七節顯示器所要顯示的資料丟到a、b…g 匯流排上，然後將1110 掃瞄信號送到四個電晶體的基極，即可顯示第一個七節顯示器；若要顯示第二個七節顯示器，同樣是將所要顯示的資料丟到a、b…g 匯流排上，然後將1101 掃瞄信號送到四個電晶體的基極，即可顯示第二個七節顯示器；若要顯示第三個七節顯示器，同樣是將所要顯示的資料丟到a、b…g 匯流排上，然後將1011掃瞄信號送到四個電晶體的基極，即可顯示第三個七節顯示器；若要顯示第四個七節顯示器，同樣是將所要顯示的資料丟到a、b…g匯流排上，然後將0111 掃瞄信號送到四個電晶體的基極，即可顯示第四個七節顯示器。掃瞄一圈後，再從頭開始掃瞄。雖然任何一個時間裡，只顯示一個七節顯示器，但只要從第一個到最後一個的掃瞄時間，不超過16ms，則因人類的視覺暫態現象，而會同時看到這幾個數字。

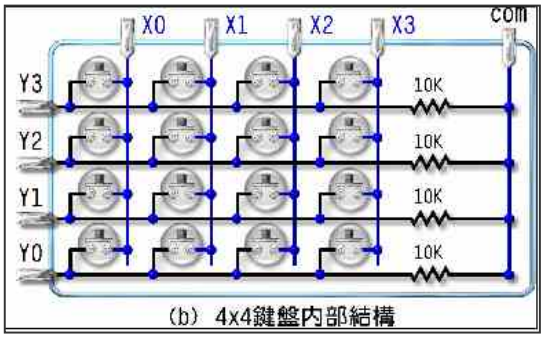
當我們使用四個七節顯示器時，每個七節顯示器都有a、b…g，線路接很麻煩！這時候可改用四個七節顯示器包裝在一起的七節顯示器模組，如下圖所示：



4x4 鍵盤掃描輸入 VHDL(1)- key1.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;
entity key1 is
port(
        clk1: in std_logic;
        y : in std_logic_vector(3 downto 0);
        x : out std_logic_vector(3 downto 0);
        s : out std_logic_vector(3 downto 0);
        seg : out std_logic_vector(0 to 6)) ;
end key1;
architecture key of key1 is
signal bcd        :        integer range 0 to 15;
signal x_code : std_logic_vector(1 downto 0);
signal keycode : std_logic_vector(3 downto 0);
begin

key_code:
process(clk1)
begin
  if clk1='1' and clk1'event then
        if y="1111" then
                if y_code="11" then
                        y_code<="00";
                else y_code<=y_code+1;
                end if;
        elsif y="1110" then
                keycode<=y_code&"00";
        elsif y="1101" then
                keycode<=y_code&"01";
        elsif y="1011" then
                keycode<=y_code&"10";
        elsif y="0111" then
                keycode<=y_code&"11";
        end if;
  end if;
end process key_code;

key_scan:
process(y_code)
begin
        case y_code is
        when "00" => x<="0111";
        when "01" => x<="1011";
        when "10" => x<="1101";
        when "11" => x<="1110";
        when others=>x<="1111";
        end case;
end process key_scan;
```

y：列掃瞄讀取信號
x：行掃描輸入信號



(b) 4x4鍵盤內部結構

偵測鍵盤是否有被按下，如果沒有則 y_code 繼續做模 4 的計數輸出

讀取鍵盤 Y 列訊號並編碼 keycode 值

讀取 y_code 訊號，作為 X 行掃瞄輸出的訊號的依據

```
decode:
process(keycode)
begin
        case keycode is
            when "0000" => bcd<=0;
            when "0001" => bcd<=1;
            when "0010" => bcd<=2;
            when "0011" => bcd<=3;
            when "0100" => bcd<=4;
            when "0101" => bcd<=5;
            when "0110" => bcd<=6;
            when "0111" => bcd<=7;
            when "1000" => bcd<=8;
            when "1001" => bcd<=9;
            when "1010" => bcd<=10;
            when "1011" => bcd<=11;
            when "1100" => bcd<=12;
            when "1101" => bcd<=13;
            when "1110" => bcd<=14;
            when "1111" => bcd<=15;
            when others => null;
            end case;
end process decode;
```

讀取 key_code 編碼值，作為七節顯示器輸出的訊號

```
--*****************************
s<="1110";    --只讓最右邊 1 顆七節顯示器亮
with bcd select
  seg <= "1111110"    when 0 ,
         "0110000"    when 1 ,
         "1101101"    when 2 ,
         "1111001"    when 3 ,
         "0110011"    when 4 ,
         "1011011"    when 5 ,
         "0011111"    when 6 ,
         "1110000"    when 7 ,
         "1111111"    when 8 ,
         "1110011"    when 9 ,
         "1000000"    when 10,
         "0100000"    when 11,
         "0010000"    when 12,
         "0001000"    when 13,
         "0000100"    when 14,
         "0000010"    when 15,
         "1001111"    when others ;
   end key    ;
```

七節顯示器之解碼輸出

4x4 鍵盤掃描編碼輸入 VHDL(2)- key2_com0.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;
entity key2_com0 is
port(
        clk1: in std_logic;
        x : out std_logic_vector(3 downto 0);
        y : in std_logic_vector(3 downto 0);
        bcd  :buffer std_logic_vector(3 downto 0));
end key2_com0;
architecture key of key2_com0 is
signal scan      :       integer range 3 downto 0;
--****************************************
component debounce
PORT(
         fs:IN STD_LOGIC;
         bounce_in:in std_logic;
         deb_out:out STD_LOGIC
        );
end component;
--****************************
begin

key_scan:
process(clk1)
        begin
          wait until clk1='1' ;
        if y="0000" then
                if scan=0 then
                        scan<=3;
                else scan<=scan-1;
                end if;
        end if;
end process key_scan;
key_x_scan:
process(clk1)
begin
        case scan is
                when 3 => x<="1000";
                when 2 => x<="0100";
                when 1 => x<="0010";
                when 0 => x<="0001";
                when others=> null;
        end case;
end process key_x_scan;
--*************************
key_read:
process(y)
begin
```

偵測鍵盤是否有被按下，如果沒有則 scan 繼續做模 4 的計數輸出

讀取 scan 訊號，作為 X 行掃瞄輸出的訊號的依據

```
if scan=3 then
      case y is
              when "1000" => bcd<="0011";
              when "0100" => bcd<="0010";
              when "0010" => bcd<="0001";
              when "0001" => bcd<="0000";
              when others => null;
            end case;
      elsif scan=2 then
        case y is
              when "1000" => bcd<="0111";
              when "0100" => bcd<="0110";
              when "0010" => bcd<="0101";
              when "0001" => bcd<="0100";
              when others => null;
            end case;
      elsif scan=1 then
        case y is
              when "1000" => bcd<="1011";
              when "0100" => bcd<="1010";
              when "0010" => bcd<="1001";
              when "0001" => bcd<="1000";
              when others => null;
            end case;
      elsif scan=0 then
        case y is
              when "1000" => bcd<="1111";
              when "0100" => bcd<="1110";
              when "0010" => bcd<="1101";
              when "0001" => bcd<="1100";
              when others => null;
            end case;
      end if ;
end process key_read;
--*****************************
   end key    ;
```

讀取 Y 列資料及編碼值輸出

此程式無 7 節顯示器顯示電路，只有將鍵盤鍵入值讀取及編碼，於下一程式中會利用 component…port map 副程式呼叫的方式將已完成之局部功能電路組合成新的完整電路