LCM_4bit_driver.vhd

```
--中文 LCM_4bit_driver(WG14432B5)
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 LCM_4bit_driver is
    port(LCM_CLK,LCM_RESET:in std_logic;    --操作速率,重置
         RS,RW:in std_logic;                --暫存器選擇,讀寫旗標輸入
         DBi:in std_logic_vector(7 downto 0);--LCM_4bit_driver 資料輸入
         DBo:out std_logic_vector(7 downto 0);--LCM_4bit_driver 資料輸出
         DB_io:inout std_logic_vector(3 downto 0);--LCM DATA BUS 介面
         RSo,RWo,Eo:out std_logic;  --LCM 暫存器選擇,讀寫,致能介面
         LCMok:out boolean          --LCM_4bit_driver 完成
        );
end entity LCM_4bit_driver;

architecture Albert of LCM_4bit_driver_delay is
    signal RWS,BF:std_logic;                    --讀寫狀態,busy
    signal LCMruns:std_logic_vector(2 downto 0);--執行狀態
    signal DBii:std_logic_vector(3 downto 0);  --內部 BUS
    signal Timeout:integer range 0 to 256;     --timeout 計時器

begin

RWo<=RWS;   --讀寫狀態輸出
DB_io<=DBii when RWS='0' else "ZZZZ";  --LCM data bus 操作

LCM_4BIT_OUT:process(LCM_CLK,LCM_RESET)
begin
    if LCM_RESET='0' then
        DBo<=(DBo'range=>'0');          --資料輸入歸零
        DBii<=DBi(7 downto 4);          --high nibble
        RSo<=RS;                        --暫存器選擇
        BF<='1';
        RWs<=RW;                        --讀寫設定
        Eo<='0';                        --LCM 禁能
        LCMok<=False;                   --未完成作業
        LCMruns<="000";                 --執行狀態由 0 開始
        Timeout<=0;                     --計時
    elsif rising_edge(LCM_CLK) then
        case LCMruns is
            when "000"=>
                Eo<='1';                --LCM 致能
                LCMruns<="001";         --執行狀態下一步
            when "001"=>
                Eo<='0';                --LCM 禁能
                if RW='1' then          --如是讀取指令
                    DBo(7 downto 4)<=DB_io;--Read Data(high nibble)
                end if;
                LCMruns<="01" & RWS;    --執行狀態下一步
```

```
            when "010"=>                --輸出
                DBii<=DBi(3 downto 0); --low nibble
                LCMruns<="011";         --執行狀態下一步
            when "011"=>
                Eo<='1';                --LCM 致能
                LCMruns<="111";         --執行狀態下一步
            when "111"=>
                if RW='1' then          --如是讀取指令
                    DBo(3 downto 0)<=DB_io;--Read Data(low nibble)
                end if;
                Eo<='0';                --LCM 禁能
                LCMruns<="110";         --執行狀態下一步
            when "110"=>                --採 delay 模式
                Timeout<=Timeout+1;     --timeout 計時
                if RS='0' and DBi=1 then--清除顯示幕指令
                    if Timeout=220 then
                        LCMruns<="101"; --執行狀態下一步
                    end if;
                elsif Timeout=2 then
                    LCMruns<="101";     --執行狀態下一步
                end if;
            when others=>               --101
                LCMok<=True;            --作業已完成
        end case;
    end if;
end process LCM_4BIT_OUT;

end Albert;
```

CH6_C_LCD_1.vhd

```
--中文 LCM 顯示(讀詩) 使用:LCM_4bit_driver
--106.12.30 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 CH6_C_LCD_1 is
port(gckp31,rstP99:in std_logic;        --系統頻率,系統 reset
      S1,S2:in std_logic;               --向上、向下按鈕
     --LCD 4bit 介面
     DB_io:inout std_logic_vector(3 downto 0);
     RSo,RWo,Eo:out std_logic
     );
end entity CH6_C_LCD_1;

architecture Albert of CH6_C_LCD_1 is
```

```
    --中文 LCM 4bit driver(WG14432B5)
    component LCM_4bit_driver is
    port(LCM_CLK,LCM_RESET:in std_logic;    --操作速率,重置
         RS,RW:in std_logic;          --暫存器選擇,讀寫旗標輸入
         DBi:in std_logic_vector(7 downto 0);
            --LCM_4bit_driver 資料輸入
         DBo:out std_logic_vector(7 downto 0);
            --LCM_4bit_driver 資料輸出
         DB_io:inout std_logic_vector(3 downto 0);
            --LCM DATA BUS 介面
         RSo,RWo,Eo:out std_logic;
            --LCM 暫存器選擇,讀寫,致能介面
         LCMok, LCM_S:out boolean    --LCM_4bit_driver 完成,錯誤旗標
         );
    end component;

    signal LCM_RESET,RS,RW:std_logic;
    --LCM_4bit_driver 重置,LCM 暫存器選擇,讀寫旗標
    signal DBi,DBo:std_logic_vector(7 downto 0);
    --LCM_4bit_driver 命令或資料輸入及輸出
    signal LCMok,LCM_S:boolean;
    --LCM_4bit_driver 完成作業旗標,錯誤信息

    signal FD:std_logic_vector(24 downto 0);--除頻器
    signal times:integer range 0 to 2047;  --計時器

    --中文 LCM 指令&資料表格式:
    --(總長,指令數,指令...資料.....)
    --英數型 LCM 4 位元界面,2 列顯示

    type LCM_T is array (0 to 20) of std_logic_vector(7 downto 0);
    constant LCM_IT:LCM_T:=(
        X"0F",X"06",          --總長,指令數
        "00101000","00101000","00101000",--4 位元界面
        "00000110","00001100","00000001",
        --ACC+1 顯示幕無移位,顯示幕 on 無游標無閃爍,清除顯示幕

        X"01",X"48",X"65",X"6C",X"6C",X"6F",X"21",X"20",
        X"20",X"20",x"20",X"20",X"20");--白臉 Hello!

    --LCM=21:第一列顯示 夜思  作者：李白
    signal LCM_21:LCM_T:=(
        X"13",X"01",          --總長,指令數
        "00000001",           --清除顯示幕
        --第 1 列顯示資料
        X"A9",X"5D",X"AB",X"E4",X"A1",X"40",X"A7",X"40",
        X"AA",X"CC",X"A1",X"47",X"A7",X"F5",X"A5",X"D5",
        X"20",X"20");         --靜夜思  作者：李白

    --LCM=22:第二列顯示 床前明月光，
    signal LCM_22:LCM_T:=(
        X"13",X"01",          --總長,指令數
        "10010000",           --設第二列 ACC 位置
```

```
        --第 2 列顯示資料
        X"A7",X"C9",X"AB",X"65",X"A9",X"FA",X"A4",X"EB",
        X"A5",X"FA",X"A1",X"41",X"20",X"20",X"20",x"20",
        X"20",X"20");--床前明月光，

    --LCM=23:第二列顯示 疑似地上霜
    signal LCM_23:LCM_T:=(
        X"13",X"01",        --總長,指令數
        "10010000",         --設第二列 ACC 位置
        --第 2 列顯示資料
        X"BA",X"C3",X"A6",X"FC",X"A6",X"61",X"A4",X"57",
        X"C1",X"F7",X"A1",X"41",X"20",X"20",X"20",x"20",
        X"20",X"20");--疑似地上霜，

    --LCM=24:第二列顯示 舉頭望明月，
    signal LCM_24:LCM_T:=(
        X"13",X"01",        --總長,指令數
        "10010000",         --設第二列 ACC 位置
        --第 2 列顯示資料
        X"C1",X"7C",X"C0",X"59",X"B1",X"E6",X"A9",X"FA",
        X"A4",X"EB",X"A1",X"41",X"20",X"20",X"20",x"20",
        X"20",X"20");--舉頭望明月，

    --LCM=25:第二列顯示 低頭思故鄉。
    signal LCM_25:LCM_T:=(
        X"13",X"01",        --總長,指令數
        "10010000",         --設第二列 ACC 位置
        --第 2 列顯示資料
        X"A7",X"43",X"C0",X"59",X"AB",X"E4",X"AC",X"47",
        X"B6",X"6D",X"A1",X"43",X"20",X"20",X"20",x"20",
        X"20",X"20");--低頭思故鄉。

    signal LCM_com_data,LCM_com_data2:LCM_T;--LCD 表格輸出
    signal LCM_INI:integer range 0 to 31;  --LCD 表格輸出指標
    signal LCMP_RESET,LN,LCMPok:std_logic;
            --LCM_P 重置,輸出列數,LCM_P 完成
    signal LCM,LCMx:integer range 0 to 7;  --LCD 輸出選項

    signal S2S,S1S:std_logic_vector(2 downto 0);--防彈跳計數器

begin

--中文 LCM----------------
LCMset: LCM_4bit_driver_delay port map(        --LCM 模組
    FD(7),LCM_RESET,RS,RW,DBi,DBo,DB_io,RSo,RWo,Eo,LCMok,LCM_S);

C_LCD_P:process(FD(18))
begin
    if rstP99='0' then              --系統重置
        LCM<=0;                     --中文 LCM 初始化
        LCMP_RESET<='0';            --LCMP 重置
    elsif rising_edge(FD(18)) then
        LCMP_RESET<='1';            --LCMP 啟動顯示
```

```
        if LCMPok='1' then
            if S1S(2)='1' then      --向上按鈕
                if LCM>1 then
                    LCM<=LCM-1;     --顯示 上一句
                end if;
            elsif S2S(2)='1' then   --向下按鈕
                if LCM<4 then
                    LCM<=LCM+1;     --顯示 下一句
                end if;
            end if;
        end if;
    end if;
end process C_LCD_P;

--中文 LCM 顯示器--
--中文 LCM 顯示器
--指令&資料表格式:
--(總長,指令數,指令...資料.....)
LCM_P:process(FD(0))
    variable SW:boolean;                --命令或資料備妥旗標
begin
    if LCM/=LCMx or LCMP_RESET='0' then
        LCMx<=LCM;                      --記錄選項
        LCM_RESET<='0';                 --LCM 重置
        LCM_INI<=2;                     --命令或資料索引設為起點
        LN<='0';                        --設定輸出 1 列
        case LCM is                     --載入選項表格
            when 0=>
                LCM_com_data<=LCM_IT;
                --LCM 初始化輸出第一列資料 Hello!
            when 1=>
                LCM_com_data<=LCM_21;   --輸出第一列資料
                LCM_com_data2<=LCM_22;  --輸出第二列資料
                LN<='1';                --設定輸出 2 列
            when 2=>
                LCM_com_data<=LCM_23;   --輸出第二列資料
            when 3=>
                LCM_com_data<=LCM_24;   --輸出第二列資料
            when others =>
                LCM_com_data<=LCM_25;   --輸出第二列資料
        end case;
        LCMPok<='0';        --取消完成信號
        SW:=False;          --命令或資料備妥旗標
    elsif rising_edge(FD(0)) then
        if SW then          --命令或資料備妥後
            LCM_RESET<='1'; --啟動 LCM_4bit_driver_delay
            SW:=False;      --重置旗標
        elsif LCM_RESET='1' then
        --LCM_4bit_driver_delay 啟動中
            if LCMok then
            --等待 LCM_4bit_driver_delay 完成傳送
                LCM_RESET<='0';         --完成後 LCM 重置
            end if;
```

```
        elsif LCM_INI<LCM_com_data(0) and
            LCM_INI<LCM_com_data'length then
            --命令或資料尚未傳完
            if LCM_INI<=(LCM_com_data(1)+1) then
            --選命令或資料暫存器
                RS<='0';    --Instruction reg
            else
                RS<='1';    --Data reg
            end if;
            RW<='0';        --LCM 寫入操作
            DBi<=LCM_com_data(LCM_INI);--載入命令或資料
            LCM_INI<=LCM_INI+1;         --命令或資料索引指到下一筆
            SW:=True;                   --命令或資料已備妥
        else
            if LN='1' then              --設定輸出 2 列
                LN<='0';                --設定輸出 2 列取消
                LCM_INI<=2;             --命令或資料索引設為起點
                LCM_com_data<=LCM_com_data2;--LCM 輸出第二列資料
            else
                LCMPok<='1';            --執行完成
            end if;
        end if;
    end if;
end process LCM_P;

--防彈跳--
process(FD(17))
begin
    --S1 防彈跳--向上按鈕
    if S1='1' then
        S1S<="000";
    elsif rising_edge(FD(17)) then
        S1S<=S1S+ not S1S(2);
    end if;
    --S1 防彈跳--向下按鈕
    if S2='1' then
        S2S<="000";
    elsif rising_edge(FD(17)) then
        S2S<=S2S+ not S2S(2);
    end if;
end process;

--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;
```

CH6_C_LCD_2.vhd

```
--中文 LCM 使用:LCM_4bit_driver
--數位電子鐘 24 小時制
--106.12.30 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 CH6_C_LCD_2 is
    port(gckp31,rstP99:in std_logic;--系統頻率,系統 reset(歸零)
         S1,S2,S3,S8:in std_logic;
         --時,分,秒遞增按鈕,設定/暫停/計時按鈕
         --LCD 4bit 介面
         DB_io:inout std_logic_vector(3 downto 0);
         RSo,RWo,Eo:out std_logic
         );
end entity CH6_C_LCD_2;

architecture Albert of CH6_C_LCD_2 is
    --中文 LCM 4bit driver(WG14432B5)
    component LCM_4bit_driver is
    port(LCM_CLK,LCM_RESET:in std_logic;    --操作速率,重置
         RS,RW:in std_logic;                --暫存器選擇,讀寫旗標輸入
         DBi:in std_logic_vector(7 downto 0); --LCM_4bit_driver 資料輸入
         DBo:out std_logic_vector(7 downto 0);--LCM_4bit_driver 資料輸出
         DB_io:inout std_logic_vector(3 downto 0);--LCM DATA BUS 介面
         RSo,RWo,Eo:out std_logic;  --LCM 暫存器選擇,讀寫,致能介面
         LCMok,LCM_S:out boolean    --LCM_4bit_driver 完成,錯誤旗標
         );
    end component;

    signal LCM_RESET,RS,RW:std_logic;
    --LCM_4bit_driver 重置,LCM 暫存器選擇,讀寫旗標
    signal DBi,DBo:std_logic_vector(7 downto 0);
    --LCM_4bit_driver 命令或資料輸入及輸出
    signal LCMok,LCM_S:boolean;
    --LCM_4bit_driver 完成作業旗標,錯誤信息

    --中文 LCM 指令&資料表格式:
    --(總長,指令數,指令...資料.....)
    --英數型 LCM 4 位元界面,2 列顯示

    type LCM_T is array (0 to 20) of std_logic_vector(7 downto 0);
    constant LCM_IT:LCM_T:=(
        X"0F",X"06",----中文型 LCM 4 位元界面
```

```
        "00101000","00101000","00101000",--4 位元界面
        "00000110","00001100","00000001",
        --ACC+1 顯示幕無移位,顯示幕 on 無游標無閃爍,清除顯示幕
        X"01",X"48",X"65",X"6C",X"6C",X"6F",X"21",X"20",
        X"20",X"20",x"20",X"20",X"20");--白臉 Hello!

    --LCM=11:第一列顯示 ☆★現在時間★☆
    signal LCM_11:LCM_T:=(
        X"13",X"01",          --總長,指令數
        "00000001",           --清除顯示幕
        --第 1 列顯示資料
        X"A1",X"B8",X"A1",X"B9",X"B2",X"7B",X"A6",X"62",
        X"AE",X"C9",X"B6",X"A1",X"A1",X"B9",X"A1",X"B8",
        X"20",X"20");--☆★現在時間★☆

    --LCM=12:第一列顯示 ▼△調整時間▽▲
    signal LCM_12:LCM_T:=(
        X"13",X"01",          --總長,指令數
        "00000001",           --清除顯示幕
        --第 2 列顯示資料
        X"A1",X"BF",X"A1",X"B5",X"BD",X"D5",X"BE",X"E3",
        X"AE",X"C9",X"B6",X"A1",X"A1",X"BE",X"A1",X"B6",
        X"20",X"20");--▼△調整時間▽▲

    --LCM=21:第二列顯示 hh:mm:ss
    signal LCM_21:LCM_T:=(
        X"13",X"01",          --總長,指令數
        "10010000",           --設第二列 ACC 位置
        --第 2 列顯示資料
        X"20",X"20",X"20",X"20",X"48",X"48",X"3A",X"4D",
        X"4D",X"3A",X"53",X"53",X"20",X"20",X"20",X"20",
        X"20",X"20");--    hh:mm:ss

    --LCM=22:第二列顯示 ＨＨ：ＭＭ：ＳＳ
    signal LCM_22:LCM_T:=(
        X"13",X"01",          --總長,指令數
        "10010000",           --設第二列 ACC 位置
        --第 2 列顯示資料
        X"A2",X"AF",X"A2",X"AF",X"A1",X"47",X"A2",X"AF",
        X"A2",X"AF",X"A1",X"47",X"A2",X"AF",X"A2",X"AF",
        X"20",X"20");--ＨＨ：ＭＭ：ＳＳ

    type N_T is array (0 to 9) of std_logic_vector(7 downto 0);
    constant N0_9_1:N_T:=(
        X"30",X"31",X"32",X"33",X"34",X"35",X"36",X"37",X"38",X"39");
        --0123456789
    constant N0_9_2:N_T:=(
        X"AF",X"B0",X"B1",X"B2",X"B3",X"B4",X"B5",X"B6",X"B7",X"B8");
        --０１２３４５６７８９

    signal LCM_com_data,LCM_com_data2:LCM_T;--LCD 表格輸出
    signal LCM_INI:integer range 0 to 31;  --LCD 表格輸出指標
    signal LCMP_RESET,LN,LCMPok:std_logic;
```

```
    --LCM_P 重置,輸出列數,LCM_P 完成
    signal LCM,LCMx:integer range 0 to 7;  --LCD 輸出選項

    signal FD:std_logic_vector(26 downto 0);   --系統除頻器
    signal Scounter:integer range 0 to 390625; --0.25 秒計時器
    signal S3S,S2S,S1S,S8S:std_logic_vector(2 downto 0);
    --防彈跳計數器
    signal H,HHH:integer range 0 to 23;    --時
    signal M,MMM,S,SSS:integer range 0 to 59;  --分,秒
    signal Ss,SS1,E_Clock_P_clk:std_logic;
    --0.5,1 秒,E_Clock_P 時脈操作
    signal MSs,MSs2:std_logic_vector(1 downto 0);--設定 on/off

begin

--中文 LCM--
LCMset: LCM_4bit_driver port map(
    FD(7),LCM_RESET,RS,RW,DBi,DBo,
    DB_io,RSo,RWo,Eo,LCMok,LCM_S); --LCM 模組

--更新顯示時間--
--0123456789
LCM_21(7)<=N0_9_1(H/10);        --時十位
LCM_21(8)<=N0_9_1(H mod 10);    --時個位
LCM_21(10)<=N0_9_1(M/10);       --分十位
LCM_21(11)<=N0_9_1(M mod 10);   --分個位
LCM_21(13)<=N0_9_1(S/10);       --秒十位
LCM_21(14)<=N0_9_1(S mod 10);   --秒個位
----0 1 2 3 4 5 6 7 8 9
LCM_22(4)<=N0_9_2(H/10);        --時十位
LCM_22(6)<=N0_9_2(H mod 10);    --時個位
LCM_22(10)<=N0_9_2(M/10);       --分十位
LCM_22(12)<=N0_9_2(M mod 10);   --分個位
LCM_22(16)<=N0_9_2(S/10);       --秒十位
LCM_22(18)<=N0_9_2(S mod 10);   --秒個位

--數位電子鐘 24 小時制--
E_Clock_P_clk<=FD(23) when MSs>0 else Ss;--E_Clock_P 時脈選擇
E_Clock_P:process(E_Clock_P_clk)
begin
    if rstP99='0' then              --系統重置,歸零
        M<=0;                       --分歸零
        S<=0;                       --秒歸零
        MSs<="00";                  --狀態切換控制
        SS1<='0';                   --秒
    elsif rising_edge(E_Clock_P_clk) then
        SS1<=not SS1;               --秒
        if S8S(2)='1' then          --狀態進行切換
            if MSs=0 or MSs=2 then  --計時轉設定 or 設定轉計時
                MSs<=MSs+1;         --切換
            end if;
        else                        --狀態轉換
            if MSs=1 or MSs=3 then  --計時轉設定 or 設定轉計時
```

```
                MSs<=MSs+1;          --轉換:轉可執行穩定狀態
                SS1<='0';            --秒重新計時
            end if;
        end if;
        if MSs>0 then                --狀態中
            if MSs=2 then            --可設定
                if S1S(2)='1' then   --調整時
                    if H=23 then
                        H<=0;
                    else
                        H<=H+1;
                    end if;
                end if;
                if S2S(2)='1' then   --調整分
                    if M=59 then
                        M<=0;
                    else
                        M<=M+1;
                    end if;
                end if;
                if S3S(2)='1' then   --調整秒
                    if S=59 then
                        S<=0;
                    else
                        S<=S+1;
                    end if;
                end if;
            end if;
        elsif SS1='1' then           --1 秒到
            if S/=59 then            --秒計時
                S<=S+1;
            else
                S<=0;
                if M/=59 then        --分計時
                    M<=M+1;
                else
                    M<=0;
                    if H/=23 then    --時計時
                        H<=H+1;
                    else
                        H<=0;
                    end if;
                end if;
            end if;
        end if;
    end if;
end process E_Clock_P;

--0.5 秒信號產生器--
S_G_P:process(FD(4))--1S:5,0.5S:4
begin
    if rstP99='0' or MSs>0 then--系統重置 or 重新計時
        Ss<='1';
```

```
        Scounter<=390625;       --0.25秒計時器預設
    elsif rising_edge(FD(4)) then--4:1562500Hz,5:781250Hz
        Scounter<=Scounter-1;   --0.25秒計時器遞減
        if Scounter=1 then      --0.25秒到
            Scounter<=390625;   --0.25秒計時器重設
            Ss<=not Ss;         --0.5秒狀態
        end if;
    end if;
end process S_G_P;

--中文LCM切換顯示
C_LCD_P:process(FD(8))
begin
    if rstP99='0' then      --系統重置
        LCM<=0;             --中文LCM初始化
        LCMP_RESET<='0';    --LCMP重置
        HHH<=0;             --時比對
        MMM<=0;             --分比對
        SSS<=0;             --秒比對
        MSs2<=(others=>'0');--模式比對
    elsif rising_edge(FD(8)) then
        LCMP_RESET<='1';    --LCMP啟動顯示
        if LCMPok='1' then  --LCM_P已完成
            if LCM=0 then   --首次切換
                LCM<=1;     --首次切換顯示:計時模式
            elsif MSs2/=MSs then--模式已改變
                MSs2<=MSs;
                if MSs(1)='0' then
                    LCM<=1; --切換顯示:計時模式
                else
                    LCM<=2; --切換顯示:調整模式
                end if;
                LCMP_RESET<='0';    --LCMP重置
            elsif HHH/=H or MMM/=M or SSS/=S then  --時間已改變
                HHH<=H; MMM<=M; SSS<=S;
                if MSs(1)='0' then
                    LCM<=3; --計時顯示
                else
                    LCM<=4; --調整顯示
                end if;
                LCMP_RESET<='0';    --LCMP重置
            end if;
        end if;
    end if;
end process C_LCD_P;

--中文LCM顯示器--
--中文LCM顯示器
--指令&資料表格式:
--(總長,指令數,指令...資料.....)
LCM_P:process(FD(0))
    variable SW:boolean;                --命令或資料備妥旗標
begin
```

```
    if LCM/=LCMx or LCMP_RESET='0' then
        LCMx<=LCM;                          --記錄選項
        LCM_RESET<='0';                     --LCM 重置
        LCM_INI<=2;                         --命令或資料索引設為起點
        LN<='0';                            --設定輸出 1 列
        case LCM is                         --載入選項表格
            when 0=>
                LCM_com_data<=LCM_IT;       --LCM 初始化輸出第一列資料 Hello!
            when 1=>
                LCM_com_data<=LCM_11;       --輸出第一列資料
                LCM_com_data2<=LCM_22;      --輸出第二列資料
                LN<='1';                    --設定輸出 2 列
            when 2=>
                LCM_com_data<=LCM_12;       --輸出第一列資料
                LCM_com_data2<=LCM_21;      --輸出第二列資料
                LN<='1';                    --設定輸出 2 列
            when 3=>
                LCM_com_data<=LCM_22;       --輸出第二列資料
            when others =>
                LCM_com_data<=LCM_21;       --輸出第二列資料
        end case;
        LCMPok<='0';                --取消完成信號
        SW:=False;                  --命令或資料備妥旗標
    elsif rising_edge(FD(0)) then
        if SW then                  --命令或資料備妥後
            LCM_RESET<='1';         --啟動 LCM_4bit_driver
            SW:=False;              --重置旗標
        elsif LCM_RESET='1' then--LCM_4bit_driver 啟動中
            if LCMok then           --等待 LCM_4bit_driver 完成傳送
                LCM_RESET<='0'; --完成後 LCM 重置
            end if;
        elsif LCM_INI<LCM_com_data(0) and
            LCM_INI<LCM_com_data'length then    --命令或資料尚未傳完
            if LCM_INI<=(LCM_com_data(1)+1) then--選命令或資料暫存器
                RS<='0';    --Instruction reg
            else
                RS<='1';    --Data reg
            end if;
            RW<='0';            --LCM 寫入操作
            DBi<=LCM_com_data(LCM_INI);--載入命令或資料
            LCM_INI<=LCM_INI+1;         --命令或資料索引指到下一筆
            SW:=True;                   --命令或資料已備妥
        else
            if LN='1' then              --設定輸出 2 列
                LN<='0';                --設定輸出 2 列取消
                LCM_INI<=2;             --命令或資料索引設為起點
                LCM_com_data<=LCM_com_data2;--LCM 輸出第二列資料
            else
                LCMPok<='1';            --執行完成
            end if;
        end if;
    end if;
end process LCM_P;
```

```
--防彈跳--
process(FD(17))
begin
    --S8 防彈跳--計時/調整
    if S8='1' then
        S8S<="000";
    elsif rising_edge(FD(17)) then
        S8S<=S8S+ not S8S(2);
    end if;
    --S1 防彈跳--時
    if S1='1' then
        S1S<="000";
    elsif rising_edge(FD(17)) then
        S1S<=S1S+ not S1S(2);
    end if;
    --S2 防彈跳--分
    if S2='1' then
        S2S<="000";
    elsif rising_edge(FD(17)) then
        S2S<=S2S+ not S2S(2);
    end if;
    --S3 防彈跳--秒
    if S3='1' then
        S3S<="000";
    elsif rising_edge(FD(17)) then
        S3S<=S3S+ not S3S(2);
    end if;
end process;

--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;
```