MCP3202_Driver.vhd

```
--MCP3202 ADC 測試
--107.01.01 版
--EP3C16Q240C8 50MHz LEs:15,408 PINs:161 ,gckp31 ,rstP99
--MCP3202: MSBF='1'(MSB 先傳)
--MCP3202_CH1_0:00(ch0),01,11(ch1),10->11
--動由 ch0 轉 ch1:接續轉換-同步輸出 ADC 值)

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 MCP3202_Driver is
    port(MCP3202_CLK_D,MCP3202_RESET:in std_logic;
         --MCP3202_Driver 驅動 clk,reset 信號
         MCP3202_AD0,MCP3202_AD1:buffer integer range 0 to 4095;
         --MCP3202 AD0,1 ch0,1 值
         MCP3202_try_N:in integer range 0 to 3;--失敗後再嘗試次數
         MCP3202_CH1_0:in std_logic_vector(1 downto 0);--輸入通道
         MCP3202_SGL_DIFF:in std_logic;          --MCP3202 SGL/DIFF
         MCP3202_Do:in std_logic;                --MCP3202 do 信號
         MCP3202_Di:out std_logic;               --MCP3202 di 信號
         MCP3202_CLK,MCP3202_CS:buffer std_logic;
         --MCP3202 clk,/cs 信號
         MCP3202_ok,MCP3202_S:buffer std_logic);
         --Driver 完成旗標 ,完成狀態
end MCP3202_Driver;

architecture Albert of MCP3202_Driver is
    signal MCP3202_tryN:integer range 0 to 3;  --失敗後再嘗試次數
    signal MCP3202Dis:std_logic_vector(2 downto 0);
    --2:MSBF+1:ODD/SIGN+0:SGL/DIFF
    signal MCP3202_ADs:std_logic_vector(11 downto 0);--轉換值收集
    signal MCP3202_Chs:std_logic_vector(1 downto 0); --ch 0,1
    signal i:integer range 0 to 31;                 --操作指標
begin

MCP3202:process(MCP3202_CLK_D,MCP3202_RESET)
begin
    if MCP3202_RESET='0' then       --未起始
        MCP3202_CS<='1';            --MCP3202 cs diable
        MCP3202_tryN<=MCP3202_try_N;--失敗後再嘗試次數(起始後無法再變)
        MCP3202_Chs<=MCP3202_CH1_0;--通道選擇(起始後無法再變)
        MCP3202_ok<='0';            --重置操作完成旗標
        MCP3202_S<='0';             --重置完成狀態
        MCP3202Dis<='1'&MCP3202_CH1_0(0)&MCP3202_SGL_DIFF;
        --2:MSBF+1:ODD/SIGN+0:SGL/DIFF(起始後無法再變)
    elsif rising_edge(MCP3202_CLK_D) then
```

```
        if MCP3202_ok='0' then      --未完成操作
            if i=17 then            --read end
                if MCP3202Dis(1)='0' then
                    MCP3202_AD0<=conv_integer(MCP3202_ADs);
                    --ch0 ADC 值
                else
                    MCP3202_AD1<=conv_integer(MCP3202_ADs);
                    --ch1 ADC 值
                end if;
                i<=0;                           --重置操作指標
                MCP3202_CS<='1';                --MCP3202 cs diable
                MCP3202Dis(1)<='1';             --ch0-->ch1
                MCP3202_ok<=not MCP3202_Chs(1) or MCP3202Dis(1);
                --自動由 ch0 轉 ch1 or 操作完成,成功完成
            elsif MCP3202_CS='1' then       --未操作
                i<=0;                           --重置操作指標
                MCP3202_Di<='1';                --start bit
                MCP3202_CS<='0';                --enable /CS
                MCP3202_CLK<='0';               --重置 MCP3202 /CLK
            else                                --操作中
                MCP3202_CLK<=not MCP3202_CLK;--MCP3202 /CLK 反向
                if MCP3202_CLK='1' then     --clk H to L:Di out
                    if i<3 then                 --MCP3202 起始階段
                        MCP3202_Di<=MCP3202Dis(i);
                        --2:MSBF+1:ODD/SIGN+0:SGL/DIFF
                        i<=i+1;                 --調整操作指標
                    end if;
                elsif i>2 then  --clk L to H:Do in --進入接收階段
                    i<=i+1;                     --調整操作指標
                    MCP3202_ADs<=MCP3202_ADs(10 downto 0)&MCP3202_Do;
                    --轉換值收集
                    if i=4 and MCP3202_Do='1' then --error
                        MCP3202_tryN<=MCP3202_tryN-1;
                        --失敗後調整再嘗試次數
                        if MCP3202_tryN=0 then  --失敗不用再試了
                            MCP3202_ok<='1';    --操作完成
                            MCP3202_S<='1';     --失敗
                        else                    --retry
                            MCP3202_CS<='1'; --MCP3202 cs diable
                        end if;
                    end if;
                end if;
            end if;
        end if;
    end if;
end process MCP3202;

end Albert;
```

MCP4822_Driver.vhd

```
--MCP4822 DAC 測試
--107.01.01 版
--EP3C16Q240C8 50MHz LEs:15,408 PINs:161 ,gckp31 ,rstP99
--MCP4822_CH_BA:00(chA),01,11(chB),10->11
-- (自動由 chA 轉 chB:接續轉換-同步輸出 DAC 值)

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
Use IEEE.std_logic_arith.all;           --引用套件

entity MCP4822_Driver is
    port(MCP4822_CLK,MCP4822_RESET:in std_logic;
         --MCP4822_Driver 驅動 clk,reset 信號
         MCP4822_DAA,MCP4822_DAB:in integer range 0 to 4095;
         --MCP4822 DAC chA0,B1 值
         MCP4822_CHB_A:in std_logic_vector(1 downto 0);--輸入通道
         MCP4822_GA_BA:in std_logic_vector(1 downto 0);--GA 0x2,1x1
         MCP4822_SHDN_BA:in std_logic_vector(1 downto 0);--/SHDN
         MCP4822_SDI,MCP4822_LDAC:out std_logic; --MCP4822 SDI 信號
         MCP4822_SCK,MCP4822_CS:buffer std_logic;
         --MCP4822 SCK,/cs 信號
         MCP4822_ok:buffer std_logic);    --Driver 完成旗標 ,完成狀態
end MCP4822_Driver;

architecture Albert of MCP4822_Driver is
    signal i:integer range 0 to 15;         --操作指標
    signal MCP4822DAx,MCP4822DAB:std_logic_vector(14 downto 0);
    --轉換值
    signal MCP4822_Chs:std_logic_vector(1 downto 0); --ch 0,1
begin

MCP4822:process(MCP4822_CLK,MCP4822_RESET)
begin
    if MCP4822_RESET='0' then                   --未起始:準備資料
        MCP4822_CS<='1';                        --MCP4822 cs diable
        MCP4822_LDAC<='1';                      --MCP4822 ldac diable
        MCP4822DAB<='0'&MCP4822_GA_BA(1)&MCP4822_SHDN_BA(1) &
                    conv_std_logic_vector(MCP4822_DAB,12);--B:DAC
        if MCP4822_CHB_A(0)='0' then
            MCP4822DAx<='0'&MCP4822_GA_BA(0)&MCP4822_SHDN_BA(0)&
                    conv_std_logic_vector(MCP4822_DAA,12);     --A:DAC
        else
            MCP4822DAx<=MCP4822DAB;             --B:DAC
        end if;
        MCP4822_Chs<=MCP4822_CHB_A;             --通道選擇
        MCP4822_ok<='0';                        --重置操作完成旗標
        i<=14;                                  --重置操作指標
```

```
    elsif rising_edge(MCP4822_CLK) then
        if MCP4822_ok='1' then              --未完成操作
            MCP4822_LDAC<='1';              --維持 AC 值
        elsif i=15 and MCP4822_SCK='1' then --write end
            MCP4822_CS<='1';                --MCP4822 cs diable
            MCP4822_Chs(0)<='1'; --chA-->chB 自動由 chA 轉 chB
            MCP4822DAx<=MCP4822DAB;         --B:DAC
            i<=14;                          --準備自動由 chA 轉 chB
            if MCP4822_Chs/="10" then       --結束
                MCP4822_LDAC<='0';          --啟動新 AC 輸出
                MCP4822_ok<='1';            --操作完成
            end if;
        elsif MCP4822_CS='1' then           --未操作
            MCP4822_SDI<=MCP4822_Chs(0);    --CH bit
            MCP4822_CS<='0';                --enable /CS
            MCP4822_SCK<='0';               --重置 MCP4822 /SCK
        else                                --操作中
            MCP4822_SCK<=not MCP4822_SCK;   --MCP4822 /SCK 反向
            if MCP4822_SCK='1' then         --clk H to L
                i<=i-1;                     --調整操作指標
                MCP4822_SDI<=MCP4822DAx(i);--SDI out
            end if;
        end if;
    end if;
end process MCP4822;

end Albert;
```

CH11_LM35_ADC_1.vhd

```
--LM35 溫度感測器測試+MCP3202 ADC+中文 LCM 顯示
--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
Use IEEE.std_logic_arith.all;           --引用套件

entity CH11_LM35_ADC_1 is
port(gckp31,rstP99:in std_logic;    --系統頻率,系統 reset
     --MCP3202
     MCP3202_Di:out std_logic;
     MCP3202_Do:in std_logic;
     MCP3202_CLK,MCP3202_CS:buffer std_logic;

     --LCD 4bit 介面
     DB_io:inout std_logic_vector(3 downto 0);
     RSo,RWo,Eo:out std_logic
```

```
     );
end entity CH11_LM35_ADC_1;

architecture Albert of CH11_LM35_ADC_1 is
    -- =ADC=
    component MCP3202_Driver is
    port(MCP3202_CLK_D,MCP3202_RESET:in std_logic;
         --MCP3202_Driver 驅動 clk,reset 信號
         MCP3202_AD0,MCP3202_AD1:buffer integer range 0 to 4095;
         --MCP3202 AD0,1 ch0,1 值
         MCP3202_try_N:in integer range 0 to 3;--失敗後再嘗試次數
         MCP3202_CH1_0:in std_logic_vector(1 downto 0);--輸入通道
         MCP3202_SGL_DIFF:in std_logic;--MCP3202 SGL/DIFF
         MCP3202_Do:in std_logic;       --MCP3202 do 信號
         MCP3202_Di:out std_logic;      --MCP3202 di 信號
         MCP3202_CLK,MCP3202_CS:buffer std_logic;
         --MCP3202 clk,/cs 信號
         MCP3202_ok,MCP3202_S:buffer std_logic);
         --Driver 完成旗標 ,完成狀態
    end component;

    signal MCP3202_CLK_D,MCP3202_RESET:std_logic;
    --MCP3202_Driver 驅動 clk,reset 信號
    signal MCP3202_AD0,MCP3202_AD1:integer range 0 to 4095;
    --MCP3202 AD 值
    signal MCP3202_try_N:integer range 0 to 3:=1; --失敗後再嘗試次數
    signal MCP3202_CH1_0:std_logic_vector(1 downto 0):="01";--ch1
    signal MCP3202_SGL_DIFF:std_logic:='1';--MCP3202 SGL/DIFF 選 SGL
    signal MCP3202_ok,MCP3202_S:std_logic;--Driver 完成旗標 ,完成狀態

    --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",--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");--Hello!
        --LCM=1:第一列顯示區");-- -=MCP3202 ADC=-

    signal LCM_1:LCM_T:=(
        X"15",X"01",          --總長,指令數
        "00000001",           --清除顯示幕
        --第 1 列顯示資料
        X"20",X"2D",X"3D",X"4D",X"43",X"50",X"33",X"32",
        X"30",X"32",X"20",X"41",X"44",X"43",X"3D",X"2D",
        X"20",X"20");-- -=MCP3202 ADC=-

        --LCM=1:第二列顯示區 LM35 測溫度為     ℃
    signal LCM_12:LCM_T:=(
        X"15",X"01",          --總長,指令數
        "10010000",           --設第二列 ACC 位置
        --第 2 列顯示資料
        X"4C",X"4D",X"33",X"35",X"B4",X"FA",X"B7",X"C5",
        X"AB",X"D7",X"AC",X"B0",X"20",X"20",X"20",X"20",
        X"A2",X"4A");--LM35 測溫度為     ℃

        --LCM=2:第一列顯示區 LM35 資料讀取失敗
    signal LCM_2:LCM_T:=(
        X"15",X"01",          --總長,指令數
        "00000001",           --清除顯示幕
        --第 1 列顯示資料
        X"4C",X"4D",X"33",X"35",X"20",X"20",X"B8",X"EA",
        X"AE",X"C6",X"C5",X"AA",X"A8",X"FA",X"A5",X"A2",
        X"B1",X"D1");--LM35 資料讀取失敗

    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 lm35T:integer range 0 to 1550;  --LM35 溫度值

begin

U2: MCP3202_Driver port map(
        FD(4),MCP3202_RESET,    --MCP3202_Driver 驅動 clk,reset 信號
        MCP3202_AD0,MCP3202_AD1,--MCP3202 AD 值
        MCP3202_try_N,          --失敗後再嘗試次數
        MCP3202_CH1_0,          --輸入通道
        MCP3202_SGL_DIFF,       --SGL/DIFF
        MCP3202_Do,             --MCP3202 do 信號
        MCP3202_Di,             --MCP3202 di 信號
        MCP3202_CLK,MCP3202_CS,--MCP3202 clk,/cs 信號
        MCP3202_ok,MCP3202_S); --Driver 完成旗標 ,完成狀態
--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

LM35P_Main:process(FD(17))
begin
    if rstP99='0' then          --系統重置
        MCP3202_RESET<='0';     --MCP3202_Driver off
        LCM<=0;                 --中文 LCM 初始化
        LCMP_RESET<='0';        --LCMP 重置
    elsif rising_edge(FD(17)) then
        LCMP_RESET<='1';        --LCMP 啟動顯示
        if LCMPok='1' then
            if MCP3202_RESET='0' then   --MCP3202_Driver 尚未啟動
                MCP3202_RESET<='1';     --ADC 資料讀取
                times<=400;             --設定計時
            elsif MCP3202_ok='1' then   --ADC 讀取結束
                times<=times-1;         --計時
                if times=0 then         --時間到
                    LCM<=1;             --中文 LCM 顯示測量值
                    LCMP_RESET<='0';    --LCMP 重置
                    MCP3202_RESET<='0';   --MCP3202_Driver 準備重新讀取資料
                elsif MCP3202_S='1' then--資料讀取失敗
                    LCM<=2;             --中文 LCM 顯示 LM35 資料讀取失敗
                end if;
            end if;
        end if;
    end if;
end process LM35P_Main;

--LM35:LCM 顯示
LM35T<=MCP3202_AD1*122/100;--5/4095*MCP3202_AD1=1.22*MCP3202_AD1
LCM_12(18)<="0011"&conv_std_logic_vector(LM35T mod 10,4);
```

```
-- 擷取小數 1 位
LCM_12(17)<=X"2E";  --.小數點
LCM_12(16)<="0011"&conv_std_logic_vector((LM35T/10) mod 10,4);
-- 擷取個位數
LCM_12(15)<="0011"&conv_std_logic_vector(LM35T/100,4);-- 擷取十位數

--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
            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_1;    --輸出第一列資料
                LCM_com_data2<=LCM_12;  --輸出第二列資料
                LN<='1';                --設定輸出 2 列
            when others =>
                LCM_com_data<=LCM_2;    --輸出第一列資料
        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;

--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;
```

CH11_Voltage_ADC_2.vhd

```
--MCP3202 ch0_1 測試+中文 LCM 顯示
--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
Use IEEE.std_logic_arith.all;       --引用套件

entity CH11_Voltage_ADC_2 is
port(gckp31,rstP99:in std_logic;    --系統頻率,系統 reset
     --MCP3202
     MCP3202_Di:out std_logic;
     MCP3202_Do:in std_logic;
     MCP3202_CLK,MCP3202_CS:buffer std_logic;

     --LCD 4bit 介面
     DB_io:inout std_logic_vector(3 downto 0);
     RSo,RWo,Eo:out std_logic;
```

```
     S1:in std_logic             --顯示選擇按鈕輸入
     );
end entity CH11_Voltage_ADC_2;

architecture Albert of CH11_Voltage_ADC_2 is
    -- =ADC=
    component MCP3202_Driver is
    port(MCP3202_CLK_D,MCP3202_RESET:in std_logic;
         --MCP3202_Driver 驅動 clk,reset 信號
         MCP3202_AD0,MCP3202_AD1:buffer integer range 0 to 4095;
         --MCP3202 AD0,1 ch0,1 值
         MCP3202_try_N:in integer range 0 to 3;--失敗後再嘗試次數
         MCP3202_CH1_0:in std_logic_vector(1 downto 0);--輸入通道
         MCP3202_SGL_DIFF:in std_logic;         --MCP3202 SGL/DIFF
         MCP3202_Do:in std_logic;               --MCP3202 do 信號
         MCP3202_Di:out std_logic;              --MCP3202 di 信號
         MCP3202_CLK,MCP3202_CS:buffer std_logic;
         --MCP3202 clk,/cs 信號
         MCP3202_ok,MCP3202_S:buffer std_logic);
         --Driver 完成旗標 ,完成狀態
    end component;

    signal MCP3202_CLK_D,MCP3202_RESET:std_logic;
            --MCP3202_Driver 驅動 clk,reset 信號
    signal MCP3202_AD0,MCP3202_AD1:integer range 0 to 4095;
            --MCP3202 AD 值
    signal MCP3202_try_N:integer range 0 to 3:=1;
            --失敗後再嘗試次數
    signal MCP3202_CH1_0:std_logic_vector(1 downto 0);
    signal MCP3202_SGL_DIFF:std_logic:='1';
            --MCP3202 SGL/DIFF 選 SGL
    signal MCP3202_ok,MCP3202_S:std_logic;
            --Driver 完成旗標 ,完成狀態

    --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_8bit_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",--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",X20"",X"20");--Hello!

    --LCM=11:第一列顯示區 --  -=MCP3202 ADC=-
    signal LCM_11:LCM_T:=(
        X"15",X"01",          --總長,指令數
        "00000001",           --清除顯示幕
        --第 1 列顯示資料
        X"20",X"2D",X"3D",X"4D",X"43",X"50",X"33",X"32",
        X"30",X"32",X"20",X"41",X"44",X"43",X"3D",X"2D",
        X"20",X"20");--  -=MCP3202 ADC=-

    --LCM=1:第二列顯示區 CH0:      CH1:
    signal LCM_12:LCM_T:=(
        X"15",X"01",          --總長,指令數
        "10010000",           --設第二列 ACC 位置
        --第 2 列顯示資料
        X"43",X"48",X"30",X"3A",X"20",X"20",X"20",X"20",
        X"20",X"20",X"43",X"48",X"31",X"3A",X"20",X"20",
        X"20",X"20");--CH0:      CH1:

    --LCM=21:第一列顯示區      --  -=電壓  測試=-
    signal LCM_21:LCM_T:=(
        X"15",X"01",          --總長,指令數
        "00000001",           --清除顯示幕
        --第 1 列顯示資料
        X"20",X"20",X"2D",X"3D",X"B9",X"71",X"C0",X"A3",
        X"20",X"20",X"B4",X"FA",X"B8",X"D5",X"3D",X"2D",
        X"20",X"20");         --  -=電壓  測試=-
```

```
    signal LCM_22:LCM_T:=(
        X"15",X"01",          --總長,指令數
        "10010000",           --設第二列 ACC 位置
        --第 2 列顯示資料
        X"43",X"48",X"30",X"3A",X"20",X"2E",X"20",X"20",
        X"20",X"20",X"43",X"48",X"31",X"3A",X"20",X"2E",
        X"20",X"20");         --CH0:      CH1:

    --LCM=31:第一列顯示區 -- -=電壓  測試=-
    signal LCM_31:LCM_T:=(
        X"15",X"01",          --總長,指令數
        "00000001",           --清除顯示幕
        --第 1 列顯示資料
        X"20",X"20",X"2D",X"3D",X"41",X"44",X"43",X"20",
        X"20",X"20",X"B9",X"71",X"C0",X"A3",X"3D",X"2D",
        X"20",X"20");         --  -=ADC   電壓=-

    signal LCM_32:LCM_T:=(
        X"15",X"01",          --總長,指令數
        "10010000",           --設第二列 ACC 位置
        --第 2 列顯示資料
        X"43",X"48",X"30",X"3A",X"20",X"20",X"20",X"20",
        X"20",X"20",X"43",X"48",X"30",X"3A",X"20",X"2E",
        X"20",X"20");         --CH0:      CH1:

    --LCM=41:第一列顯示區 -- -=電壓  測試=-
    signal LCM_41:LCM_T:=(
        X"15",X"01",          --總長,指令數
        "00000001",           --清除顯示幕
        --第 1 列顯示資料
        X"20",X"20",X"2D",X"3D",X"41",X"44",X"43",X"20",
        X"20",X"20",X"B9",X"71",X"C0",X"A3",X"3D",X"2D",
        X"20",X"20");         --  -=ADC   電壓=-

    signal LCM_42:LCM_T:=(
        X"15",X"01",          --總長,指令數
        "10010000",           --設第二列 ACC 位置
        --第 2 列顯示資料
        X"43",X"48",X"31",X"3A",X"20",X"20",X"20",X"20",
        X"20",X"20",X"43",X"48",X"31",X"3A",X"20",X"2E",
        X"20",X"20");         --CH0:      CH1:

    --LCM=2:第一列顯示區      資料讀取失敗
    signal LCM_5:LCM_T:=(
        X"15",X"01",          --總長,指令數
        "00000001",           --清除顯示幕
        --第 1 列顯示資料
        X"20",X"20",X"20",X"20",X"20",X"20",X"B8",X"EA",
```

```
        X"AE",X"C6",X"C5",X"AA",X"A8",X"FA",X"A5",X"A2",
        X"B1",X"D1");       --資料讀取失敗

    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 LCM_DM:integer range 0 to 3;
    signal S1S:std_logic_vector(2 downto 0);--防彈跳計數器
    signal CH0ADC1,CH0ADC2,CH0ADC3,CH0ADC4:std_logic_vector(7 downto 0);
    --ADC 顯示轉換值
    signal CH1ADC1,CH1ADC2,CH1ADC3,CH1ADC4:std_logic_vector(7 downto 0);
    --ADC 顯示轉換值
    signal CH0V,CH1V:integer range 0 to 511;--電壓值
    signal CH0V3,CH0V2,CH0V1:std_logic_vector(7 downto 0);
    --電壓值顯示轉換值
    signal CH1V3,CH1V2,CH1V1:std_logic_vector(7 downto 0);
    --電壓值顯示轉換值

begin

U2: MCP3202_Driver port map(
            FD(4),MCP3202_RESET,--MCP3202_Driver 驅動 clk,reset 信號
            MCP3202_AD0,MCP3202_AD1,    --MCP3202 AD 值
            MCP3202_try_N,              --失敗後再嘗試次數
            MCP3202_CH1_0,              --輸入通道
            MCP3202_SGL_DIFF,           --SGL/DIFF
            MCP3202_Do,                 --MCP3202 do 信號
            MCP3202_Di,                 --MCP3202 di 信號
            MCP3202_CLK,MCP3202_CS,     --MCP3202 clk,/cs 信號
            MCP3202_ok,MCP3202_S);      --Driver 完成旗標 ,完成狀態
--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

Voltage_Main:process(FD(17))
begin
    if rstP99='0' then          --系統重置
        MCP3202_RESET<='0';     --MCP3202_driver 重置
        LCM<=0;                 --中文 LCM 初始化
        LCMP_RESET<='0';        --LCMP 重置
        MCP3202_CH1_0<="10";    --CH0->CH1 自動轉換同步輸出
        --MCP3202_CH1_0<="00"; --CH0,CH1 輪流轉換輪流輸出
    elsif rising_edge(FD(17)) then
        LCMP_RESET<='1';        --LCMP 啟動顯示
```

```
        if LCMPok='1' then      --LCM 顯示完成
            if MCP3202_RESET='0' then  --MCP3202_driver 尚未啟動
                MCP3202_RESET<='1';     --重新讀取資料
                times<=80;              --設定計時
            elsif MCP3202_ok='1' then  --讀取結束
                times<=times-1;         --計時
                if times=0 then         --到
                    LCM<=1+LCM_DM;      --中文 LCM 顯示測量值
                    LCMP_RESET<='0';    --LCMP 重置
                    MCP3202_RESET<='0'; --準備重新讀取資料
                    --MCP3202_CH1_0(0)<=not MCP3202_CH1_0(0);
                    --CH0,CH1 輪流轉換輪流輸出
                elsif MCP3202_S='1' then--資料讀取失敗
                    LCM<=5;             --中文 LCM 顯示 資料讀取失敗
                end if;
            end if;
        end if;
    end if;
end process Voltage_Main;

--按鈕操作--
process(FD(18))
begin
    if rstP99='0' then
        LCM_DM<=0;  --顯示選擇 0
    elsif rising_edge(FD(18)) then
        if S1S(1)='1' then
            LCM_DM<=LCM_DM+1;--顯示選擇
        end if;

    end if;
end process;

--ADC 轉換顯示
CH0ADC1<="0011"&conv_std_logic_vector(MCP3202_AD0 mod 10,4);
-- 擷取個位數
CH0ADC2<="0011"&conv_std_logic_vector((MCP3202_AD0/10)mod 10,4);
-- 擷取十位數
CH0ADC3<="0011"&conv_std_logic_vector((MCP3202_AD0/100) mod 10,4);
-- 擷取百位數
CH0ADC4<="0011"&conv_std_logic_vector(MCP3202_AD0/1000,4);
-- 擷取千位數
CH1ADC1<="0011"&conv_std_logic_vector(MCP3202_AD1 mod 10,4);
-- 擷取個位數
CH1ADC2<="0011"&conv_std_logic_vector((MCP3202_AD1/10)mod 10,4);
-- 擷取十位數
CH1ADC3<="0011"&conv_std_logic_vector((MCP3202_AD1/100) mod 10,4);
-- 擷取百位數
CH1ADC4<="0011"&conv_std_logic_vector(MCP3202_AD1/1000,4);
```

```
-- 擷取千位數

--電壓值轉換顯示
CH0V<=(MCP3202_AD0*122+500)/1000;
CH0V3<="0011" & conv_std_logic_vector(CH0V/100,4);
--整數部分
CH0V2<="0011" & conv_std_logic_vector((CH0V/10)mod 10,4);
--小數第 1 位部分
CH0V1<="0011" & conv_std_logic_vector(CH0V mod 10,4);
--小數第 2 位部分
CH1V<=(MCP3202_AD1*122+500)/1000;
CH1V3<="0011" & conv_std_logic_vector(CH1V/100,4);
--整數部分
CH1V2<="0011" & conv_std_logic_vector((CH1V/10)mod 10,4);
--小數第 1 位部分
CH1V1<="0011" & conv_std_logic_vector(CH1V mod 10,4);
--小數第 2 位部分

--LCM 顯示 CH0:ADC CH1:ADC
LCM_12(7)<=CH0ADC4;     --千位數
LCM_12(8)<=CH0ADC3;     --百位數
LCM_12(9)<=CH0ADC2;     --十位數
LCM_12(10)<=CH0ADC1;    --個位數

LCM_12(17)<=CH1ADC4;    --千位數
LCM_12(18)<=CH1ADC3;    --百位數
LCM_12(19)<=CH1ADC2;    --十位數
LCM_12(20)<=CH1ADC1;    --個位數

--LCM 顯示 CH0:V CH1:V
LCM_22(7)<=CH0V3;       --整數部分
LCM_22(9)<=CH0V2;       --小數第 1 位部分
LCM_22(10)<=CH0V1;      --小數第 2 位部分

LCM_22(17)<=CH1V3;      --整數部分
LCM_22(19)<=CH1V2;      --小數第 1 位部分
LCM_22(20)<=CH1V1;      --小數第 2 位部分

--LCM 顯示 CH0:ADC V
LCM_32(7)<=CH0ADC4;     --千位數
LCM_32(8)<=CH0ADC3;     --百位數
LCM_32(9)<=CH0ADC2;     --十位數
LCM_32(10)<=CH0ADC1;    --個位數

LCM_32(17)<=CH0V3;      --整數部分
LCM_32(19)<=CH0V2;      --小數第 1 位部分
LCM_32(20)<=CH0V1;      --小數第 2 位部分

--LCM 顯示 CH1:ADC V
```

```
LCM_42(7)<=CH1ADC4;     --千位數
LCM_42(8)<=CH1ADC3;     --百位數
LCM_42(9)<=CH1ADC2;     --十位數
LCM_42(10)<=CH1ADC1;    --個位數

LCM_42(17)<=CH1V3;      --整數部分
LCM_42(19)<=CH1V2;      --小數第 1 位部分
LCM_42(20)<=CH1V1;      --小數第 2 位部分

--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
            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_12;  --輸出第二列資料
                LN<='1';                --設定輸出 2 列
            when 2=>
                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 3=>
                LCM_com_data<=LCM_31;   --輸出第一列資料
                LCM_com_data2<=LCM_32;  --輸出第二列資料
                LN<='1';                --設定輸出 2 列
            when 4=>
                LCM_com_data<=LCM_41;   --輸出第一列資料
                LCM_com_data2<=LCM_42;  --輸出第二列資料
                LN<='1';                --設定輸出 2 列
            when others =>
                LCM_com_data<=LCM_5;    --輸出第一列資料
        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--啟動按鈕
    if S1='1' then
        S1S<="000";
    elsif rising_edge(FD(17)) then
        S1S<=S1S+ not S1S(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;
```

CH11_ADC_to_DAC_3.vhd

```
--MCP3202 ch0_1->MCP4822 chA_B 測試+中文 LCM 顯示
--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
Use IEEE.std_logic_arith.all;          --引用套件

entity CH11_ADC_to_DAC_3 is
port(gckp31,rstP99:in std_logic;       --系統頻率,系統 reset
     --MCP3202 ADC
     MCP3202_Di:out std_logic;
     MCP3202_Do:in std_logic;
     MCP3202_CLK,MCP3202_CS:buffer std_logic;

     --MCP4822 DAC
     MCP4822_SDI,MCP4822_LDAC:out std_logic;--MCP4822 SDI,LDAC 信號
     MCP4822_SCK,MCP4822_CS:buffer std_logic;--MCP4822 SCK,/cs 信號

     --LCD 4bit 介面
     DB_io:inout std_logic_vector(3 downto 0);
     RSo,RWo,Eo:out std_logic

     );
end entity CH11_ADC_to_DAC_3;

architecture Albert of CH11_ADC_to_DAC_3 is
    --MCP3202 ADC--
    component MCP3202_Driver is
    port(MCP3202_CLK_D,MCP3202_RESET:in std_logic;
         --MCP3202_Driver 驅動 clk,reset 信號
         MCP3202_AD0,MCP3202_AD1:buffer integer range 0 to 4095;
         --MCP3202 AD0,1 ch0,1 值
         MCP3202_try_N:in integer range 0 to 3;--失敗後再嘗試次數
         MCP3202_CH1_0:in std_logic_vector(1 downto 0);--輸入通道
         MCP3202_SGL_DIFF:in std_logic;     --MCP3202 SGL/DIFF
         MCP3202_Do:in std_logic;           --MCP3202 do 信號
         MCP3202_Di:out std_logic;          --MCP3202 di 信號
         MCP3202_CLK,MCP3202_CS:buffer std_logic;
```

```
        --MCP3202 clk,/cs 信號
        MCP3202_ok,MCP3202_S:buffer std_logic);
        --Driver 完成旗標 ,完成狀態
    end component;

    signal MCP3202_CLK_D,MCP3202_RESET:std_logic;
    --MCP3202_Driver 驅動 clk,reset 信號
    signal MCP3202_AD0,MCP3202_AD1:integer range 0 to 4095;
    --MCP3202 AD 值
    signal MCP3202_try_N:integer range 0 to 3:=1;--失敗後再嘗試次數
    signal MCP3202_CH1_0:std_logic_vector(1 downto 0):="01";--ch1
    signal MCP3202_SGL_DIFF:std_logic:='1';--MCP3202 SGL/DIFF 選 SGL
    signal MCP3202_ok,MCP3202_S:std_logic;    --Driver 完成旗標 ,完成狀態

    --MCP4822 DAC--------------
    component MCP4822_Driver is
    port(MCP4822_CLK,MCP4822_RESET:in std_logic;
        --MCP4822_Driver 驅動 clk,reset 信號
        MCP4822_DAA,MCP4822_DAB:in integer range 0 to 4095;
        --MCP4822 DAC chA0,B1 值
        MCP4822_CHB_A:in std_logic_vector(1 downto 0);--輸入通道
        MCP4822_GA_BA:in std_logic_vector(1 downto 0);--/GA 0x2,1x1
        MCP4822_SHDN_BA:in std_logic_vector(1 downto 0);--/SHDN
        MCP4822_SDI,MCP4822_LDAC:out std_logic;--MCP4822 SDI,LDAC 信號
        MCP4822_SCK,MCP4822_CS:buffer std_logic;--MCP4822 SCK,/cs 信號
        MCP4822_ok:buffer std_logic);            --Driver 完成旗標
    end component;

    signal MCP4822_CLK,MCP4822_RESET:std_logic;
    --MCP4822_Driver 驅動 clk,reset 信號
    signal MCP4822_DAA,MCP4822_DAB:integer range 0 to 4095;
    --MCP4822 DAC chA0,B1 值
    signal MCP4822_CHB_A:std_logic_vector(1 downto 0);--輸入通道
    signal MCP4822_GA_BA:std_logic_vector(1 downto 0);--GA 0x2,1x1
    signal MCP4822_SHDN_BA:std_logic_vector(1 downto 0);--/SHDN
    signal MCP4822_ok:std_logic;          --Driver 完成旗標

    --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 FS:integer range 0 to 31;        --頻率選擇
    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",--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");--Hello!

    --LCM=1:第一列顯示區");-- -=MCP3202 ADC=-
    signal LCM_1:LCM_T:=(
            X"15",X"01",              --總長,指令數
            "00000001",               --清除顯示幕
            --第 1 列顯示資料
            X"20",X"2D",X"3D",X"4D",X"43",X"50",X"33",X"32",
            X"30",X"32",X"20",X"41",X"44",X"43",X"3D",X"2D",
            X"20",X"20");-- -=MCP3202 ADC=-

    --LCM=1:第二列顯示區 CH0:      CH1:
    signal LCM_12:LCM_T:=(
            X"15",X"01",              --總長,指令數
            "10010000",               --設第二列 ACC 位置
            --第 2 列顯示資料
            X"43",X"48",X"30",X"3A",X"20",X"20",X"20",X"20",
            X"20",X"20",X"43",X"48",X"31",X"3A",X"20",X"20",
            X"20",X"20");--CH0:      CH1:

    --LCM=2:第一列顯示區 資料讀取失敗
    signal LCM_2:LCM_T:=(
            X"15",X"01",              --總長,指令數
            "00000001",               --清除顯示幕
            --第 1 列顯示資料
```

```
            X"20",X"20",X"20",X"20",X"20",X"20",X"B8",X"EA",
            X"AE",X"C6",X"C5",X"AA",X"A8",X"FA",X"A5",X"A2",
            X"AE",X"B1",X"D1");--

    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

begin

U1: MCP4822_Driver port map(
        FD(0),MCP4822_RESET, --MCP4822_Driver 驅動 clk,reset 信號
        MCP4822_DAA,MCP4822_DAB,--MCP4822 DAC chA0,B1 值
        MCP4822_CHB_A,          --輸入通道
        MCP4822_GA_BA,          --GA 0x2,1x1
        MCP4822_SHDN_BA,        --/SHDN
        MCP4822_SDI,MCP4822_LDAC,--MCP4822 SDI,LDAC 信號
        MCP4822_SCK,MCP4822_CS,--MCP4822 SCK,/cs 信號
        MCP4822_ok);            --Driver 完成旗標

U2: MCP3202_Driver port map(
        FD(4),MCP3202_RESET,    --MCP3202_Driver 驅動 clk,reset 信號
        MCP3202_AD0,MCP3202_AD1,--MCP3202 AD 值
        MCP3202_try_N,          --失敗後再嘗試次數
        MCP3202_CH1_0,          --輸入通道
        MCP3202_SGL_DIFF,       --SGL/DIFF
        MCP3202_Do,             --MCP3202 do 信號
        MCP3202_Di,             --MCP3202 di 信號
        MCP3202_CLK,MCP3202_CS,--MCP3202 clk,/cs 信號
        MCP3202_ok,MCP3202_S); --Driver 完成旗標 ,完成狀態

--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

MCP4822_DAA<=MCP3202_AD0;--CH0:ADC to DAC
MCP4822_DAB<=MCP3202_AD1;--CH1:ADC to DAC

ADC_DAC_Main:process(FD(17))
begin
    if rstP99='0' then --系統重置
        MCP3202_RESET<='0';     --MCP3202_driver 重置
        LCM<=0;                 --中文 LCM 初始化
        LCMP_RESET<='0';        --LCMP 重置
        MCP3202_CH1_0<="10";    --CH0->CH1 自動轉換同步輸出
        --MCP3202_CH1_0<="00"; --CH0,CH1 輪流轉換輪流輸出
```

```
        MCP4822_RESET<='0';
        MCP4822_CHB_A<="10";    --CHA->CHB 自動轉換同步輸出
        MCP4822_GA_BA<="11";    --A:x1 B:x1
        MCP4822_SHDN_BA<="11"; --/SHUTDOWN off
        FS<=0;                  --頻率選擇
    elsif rising_edge(FD(FS)) then
        LCMP_RESET<='1';        --LCMP 啟動顯示
        if LCMPok='1' then      --LCM 顯示完成
            if MCP3202_RESET='0' then  --MCP3202_driver 尚未啟動
                MCP3202_RESET<='1';    --重新讀取資料
                times<=40;             --設定計時
                FS<=0;                 --頻率選擇
            elsif MCP3202_ok='1' then  --讀取結束
                if MCP4822_RESET='0' then
                    MCP4822_RESET<='1'; --啟動 DAC 轉換
                elsif MCP4822_ok='1' then
                    FS<=17;             --頻率選擇
                    times<=times-1;     --計時
                    if times=0 then     --時間到
                        LCM<=1;         --中文 LCM 顯示測量值
                        LCMP_RESET<='0';--LCMP 重置
                        MCP3202_RESET<='0'; --準備重新讀取資料
                        --MCP3202_CH1_0(0)<=not MCP3202_CH1_0(0);
                        --CH0,CH1 輪流轉換輪流輸出
                        MCP4822_RESET<='0';
                    elsif MCP3202_S='1' then--資料讀取失敗
                        LCM<=2;         --中文 LCM 顯示 資料讀取失敗
                    end if;
                end if;
            end if;
        end if;
end process ADC_DAC_Main;

--LCM 顯示
LCM_12(10)<="0011"&conv_std_logic_vector(MCP3202_AD0 mod 10,4);
-- 擷取個位數
LCM_12(9)<="0011"&conv_std_logic_vector((MCP3202_AD0/10)mod 10,4);
-- 擷取十位數
LCM_12(8)<="0011"&conv_std_logic_vector((MCP3202_AD0/100) mod 10,4);
-- 擷取百位數
LCM_12(7)<="0011"&conv_std_logic_vector(MCP3202_AD0/1000,4);
-- 擷取千位數

LCM_12(20)<="0011"&conv_std_logic_vector(MCP3202_AD1 mod 10,4);
-- 擷取個位數
LCM_12(19)<="0011"&conv_std_logic_vector((MCP3202_AD1/10)mod 10,4);
-- 擷取十位數
LCM_12(18)<="0011"&conv_std_logic_vector((MCP3202_AD1/100) mod 10,4);
```

```
-- 擷取百位數
LCM_12(17)<="0011"&conv_std_logic_vector(MCP3202_AD1/1000,4);
-- 擷取千位數

--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
            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_1;--輸出第一列資料
                LCM_com_data2<=LCM_12;--輸出第二列資料
                LN<='1';              --設定輸出 2 列
            when others =>
                LCM_com_data<=LCM_2;--輸出第一列資料
        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;

--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;
```