MG90S_Driver.vhd

```
--MG90S 測試
--107.01.01 版
--EP3C16Q240C8 50MHz LEs:15,408 PINs:161 ,gckp31 ,rstP9

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 MG90S_Driver is
    port(MG90S_CLK,MG90S_RESET:in std_logic;
         --MG90S_Driver 驅動 clk(25MHz),reset 信號
         MG90S_dir:in std_logic;                  --轉動方向
         MG90S_deg:in integer range 0 to 90;      --轉動角度
         MG90S_o:out std_logic);                  --Driver 輸出
end entity MG90S_Driver;

architecture Albert of MG90S_Driver is
    signal MG90Servd:integer range 0 to 25010;    --角度換算值
    signal MG90Servs:integer range 0 to 63000;    --servo pwm 比率值
    signal MG90Serv:integer range 0 to  500000;   --servo pwm 產生器

begin

--角度換算值--0~90--
MG90Servd<=2778*MG90S_deg/10;  --角度換算值+-25000
MG90Servs<=37500+MG90Servd when MG90S_dir='0'
            else 37500-MG90Servd;

--servo pwm 產生器--
MG90S_o<='1' when MG90Serv<MG90Servs and MG90S_RESET='1'
             else '0';

--50Hz 產生器(20ms)
MG90S:process(MG90S_CLK,MG90S_RESET)
begin
    if MG90S_RESET='0' then
        MG90Serv<=0;
    elsif rising_edge(MG90S_CLK) then
        --20ms
        MG90Serv<=MG90Serv+1;
        if MG90Serv=499999 then--f=50Hz, T=20ms
            MG90Serv<=0;
        end if;
    end if;
end process MG90S;

end Albert;
```

CH12_MG90S_1.vhd

```
--搖桿或滑桿(電位計)操控 MCP3202 ch0_1 +MG90S 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 CH12_MG90S_1 is
port(gckp31,rstP99:in std_logic;       --系統頻率,系統 reset

     --MG90S--
     MG90S_o0:out std_logic;
     MG90S_o1:out std_logic;

     --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 CH12_MG90S_1;

architecture Albert of CH12_MG90S_1 is

    --MG90S--
    component MG90S_Driver is
    port(MG90S_CLK,MG90S_RESET:in std_logic;
         --MG90S_Driver 驅動 clk(25MHz),reset 信號
         MG90S_dir:in std_logic;                --轉動方向
         MG90S_deg:in integer range 0 to 90;    --轉動角度
         MG90S_o:out std_logic);                --Driver 輸出
    end component;
    signal MG90S_CLK,MG90S_RESET:std_logic;
    --MG90S_Driver 驅動 clk(25MHz),reset 信號
    signal MG90S_dir0,MG90S_dir1:std_logic;        --轉動方向
    signal MG90S_deg0,MG90S_deg1:integer range 0 to 90;--轉動角度

-- ==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_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
    signal MG90S_degs0,MG90S_degs1:integer range 0 to 2048;
            --轉動角度

    --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"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

begin

U1: MG90S_Driver port map(
        FD(0),MG90S_RESET,  --MG90S_Driver 驅動 clk(25MHz),reset 信號
        MG90S_dir0,         --轉動方向
        MG90S_deg0,         --轉動角度
        MG90S_o0);          --Driver 輸出

U12: MG90S_Driver port map(
        FD(0),MG90S_RESET,  --MG90S_Driver 驅動 clk(25MHz),reset 信號
        MG90S_dir1,         --轉動方向
        MG90S_deg1,         --轉動角度
```

```
        MG90S_o1);          --Driver 輸出

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

--ADC 轉角度換算--
MG90S_dir0<='0' when MCP3202_AD0>=2048 else '1';--方向
MG90S_degs0<=MCP3202_AD0-2048 when MG90S_dir0='0'
                else 2048-MCP3202_AD0;
MG90S_deg0<=MG90S_degs0*10/227;                 --0~90

MG90S_dir1<='0' when MCP3202_AD1>=2048 else '1';--方向
MG90S_degs1<=MCP3202_AD1-2048 when MG90S_dir1='0'
                else 2048-MCP3202_AD1;
MG90S_deg1<=MG90S_degs1*10/227;                 --0~90

MG90S_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
        MG90S_RESET<='0';
    elsif rising_edge(FD(17)) then
        LCMP_RESET<='1';        --LCMP 啟動顯示
        MG90S_RESET<='1';
        if LCMPok='1' then      --LCM 顯示完成
            MG90S_RESET<='1';
            if MCP3202_RESET='0' then--MCP3202_driver 尚未啟動
                MCP3202_RESET<='1'; --重新讀取資料
                times<=20;          --設定計時
            elsif MCP3202_ok='1' then   --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'; --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<=2;              --中文 LCM 顯示 資料讀取失敗
                end if;
            end if;
        end if;
    end if;
end process MG90S_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;
```

MG90S_Driver2.vhd

```
--MG90S 測試
--107.01.01 版
--EP3C16Q240C8 50MHz LEs:15,408 PINs:161 ,gckp31 ,rstP9

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 MG90S_Driver2 is
    port(MG90S_CLK,MG90S_RESET:in std_logic;
        --MG90S_Driver 驅動 clk(25MHz),reset 信號
         MG90S_deg:in integer range 0 to 180;  --轉動角度
         MG90S_o:out std_logic);               --Driver 輸出
end entity MG90S_Driver2;

architecture Albert of MG90S_Driver2 is
    signal MG90Servd:integer range 0 to 50010; --角度換算值
    signal MG90Servs:integer range 0 to 63000; --servo pwm 比率值
    signal MG90Serv:integer range 0 to  500000;--servo pwm 產生器

begin

--角度換算值--0~180--
MG90Servd<=2778*MG90S_deg/10;  --角度換算值 0~50000
MG90Servs<=12500+MG90Servd;

--servo pwm 產生器--
MG90S_o<='1' when MG90Serv<MG90Servs and MG90S_RESET='1' else '0';

--50Hz 產生器
MG90S:process(MG90S_CLK,MG90S_RESET)
begin
    if MG90S_RESET='0' then
        MG90Serv<=0;
    elsif rising_edge(MG90S_CLK) then
        --20ms
        MG90Serv<=MG90Serv+1;
        if MG90Serv=499999 then--50Hz
            MG90Serv<=0;
        end if;
    end if;
end process MG90S;

end Albert;
```

CH12_MG90S_2.vhd

```
--旋轉編碼器+MG90S 測試
--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 CH12_MG90S_2 is
port(gckp31,rstP99:in std_logic;--系統頻率,系統 reset

     --MG90S--
     MG90S_o0:out std_logic;
     MG90S_o1:out std_logic;

     APi,BPi,PBi:in std_logic;   --旋轉編碼器
     LED1_2:buffer std_logic_vector(1 downto 0)--LED 顯示
     );
end entity CH12_MG90S_2;

architecture Albert of CH12_MG90S_2 is
    --MG90S--
    component MG90S_Driver2 is
    port(MG90S_CLK,MG90S_RESET:in std_logic;
         --MG90S_Driver 驅動 clk(25MHz),reset 信號
         MG90S_deg:in integer range 0 to 180;  --轉動角度
         MG90S_o:out std_logic);               --Driver 輸出
    end component;
    signal MG90S_CLK,MG90S_RESET:std_logic;
    --MG90S_Driver 驅動 clk(25MHz),reset 信號
    signal MG90S_deg0,MG90S_deg1:integer range 0 to 180:=90;
    --轉動角度

    signal FD:std_logic_vector(24 downto 0);   --除頻器
    signal times:integer range 0 to 2047;      --計時器
    signal APic,BPic,PBic:std_logic_vector(2 downto 0):="000";
    --防彈跳計數器
    signal clrPC,set90,HV:std_logic;    --清除按鈕紀錄,設 90 度,軸向
    signal PC:integer range 0 to 3;         --按鈕紀錄

begin

U1: MG90S_Driver2 port map(
        FD(0),MG90S_RESET,--MG90S_Driver 驅動 clk(25MHz),reset 信號
        MG90S_deg0,      --轉動角度
        MG90S_o0);       --Driver 輸出

U12: MG90S_Driver2 port map(
        FD(0),MG90S_RESET,--MG90S_Driver 驅動 clk(25MHz),reset 信號
        MG90S_deg1,      --轉動角度
        MG90S_o1);       --Driver 輸出

LED1_2<=HV & not HV;

--旋轉編碼器按鈕監控--
--軸向變換,設 90 度
process(FD(17),rstP99)
begin
    if rstP99='0' then
```

```
        HV<='0';                    --由 0 開始
        set90<='0';                 --不設 90 度
        clrPC<='0';                 --不清除按鈕紀錄
        times<=0;                   --計時歸零
        MG90S_RESET<='0';           --重置 MG90S_Driver2
    elsif rising_edge(FD(17)) then -- 偵測到 UD 信號的升緣時
        MG90S_RESET<='1';           --重啟 MG90S_Driver2
        if PC/=0 then               --有按
            times<=times+1;         --計時
            if times=75 then        --計時到
                if PC=1 then        --單按
                    HV<=not HV; --切換軸向
                else                --雙按
                    set90<='1'; --設 90 度
                end if;
                clrPC<='1';         --清除按鈕紀錄
            end if;
        else
            times<=0;               --計時歸零
            set90<='0';             --清除設 90 度
            clrPC<='0';             --不清除按鈕紀錄
        end if;
    end if;
end process;

--旋轉編碼器按鈕介面電--
--單按 雙按
process(PBic(2),rstP99,clrPC)
begin
    if rstP99='0' or clrPC='1' then
        PC<=0;  --按鈕紀錄 0
    elsif rising_edge(PBic(2)) then-- 偵測到 UD 信號的升緣時
        if PC<2 then
            PC<=PC+1;               --按鈕紀錄
        end if;
    end if;
end process;

--旋轉編碼器旋轉介面電路--
--角度變換 0~180
EncoderInterface:process(APi,PBi,rstP99,set90)
begin
    if rstP99='0' or set90='1' then
        MG90S_deg0<=90;
        MG90S_deg1<=90;
    elsif rising_edge(APic(2)) then     --偵測到 UD 信號的升緣時
        if HV='0' then
            if BPi='1' then             --右旋
                if MG90S_deg0<180 then
                    MG90S_deg0<=MG90S_deg0+1;--加 1 度
                end if;
            else                        --左旋
                if MG90S_deg0>0 then
```

```
                    MG90S_deg0<=MG90S_deg0-1;--減 1 度
                end if;
            end if;
        else
            if BPi='0' then             --左旋
                if MG90S_deg1<180 then
                    MG90S_deg1<=MG90S_deg1+1;--加 1 度
                end if;
            else                        --右旋
                if MG90S_deg1>0 then
                    MG90S_deg1<=MG90S_deg1-1;--減 1 度
                end if;
            end if;
        end if;
    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
        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
        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
        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;
```