RS232_T1.vhd

```
--RS232TX
Library IEEE;
Use IEEE.std_logic_1164.all;
Use IEEE.std_logic_unsigned.all;

Entity RS232_T1 is
    Port(clk,Reset:in std_logic;--clk:25MHz
         DL:in std_logic_vector(1 downto 0);
         --00:5,01:6,10:7,11:8 Bit
         ParityN:in std_logic_vector(2 downto 0);
         --0xx:None,100:Even,101:Odd,110:Space,111:Mark
         StopN:in std_logic_vector(1 downto 0);
         --0x:1Bit,10:2Bit,11:1.5Bit
         F_Set:in std_logic_vector(2 downto 0);
         Status_s:out std_logic_vector(1 downto 0);
         TX_W:in std_logic;
         TXData:in std_logic_vector(7 downto 0);
         TX:out std_logic);
end RS232_T1;

Architecture Albert of RS232_T1 is
signal StopNn:std_logic_vector(2 downto 0);
signal Tx_B_Empty,Tx_B_Clr,TxO_W:std_logic;

signal Tx_f,T_Half_f,TX_P_NEOSM:std_logic;
signal TXDs_Bf,TXD2_Bf:std_logic_vector(7 downto 0);
signal Tsend_DLN,DLN:std_logic_vector(3 downto 0);
signal Tx_s:std_logic_vector(2 downto 0);
signal TX_BaudRate:integer range 0 to 20832;
signal BaudRate1234:std_logic_vector(1 downto 0);

begin
Status_s<=Tx_B_Empty & TxO_W;

TxWP:process(TX_W,Reset)
begin
if reset='0' or Tx_B_Clr='1' then
    Tx_B_Empty<='0';
    TxO_W<='0';
elsif rising_edge(Tx_W) then
    TXD2_Bf<=TXData;
    Tx_B_Empty<='1';        --Tx_B_Empty='1'表示已有資料寫入(尚未傳出)
    TxO_W<=Tx_B_Empty;      --TxO_W='1'表示資料未傳出又寫入資料(覆寫)
end if;
end process TxWP;

TxP:process(Tx_f,Reset)
begin
if Reset='0' then
    Tx_s<="000";
```

```
    TX<='1';
    Tx_B_Clr<='0';
elsif rising_edge(Tx_f) then
    if Tx_s=0 and Tx_B_Empty='1' then--start bit
        TXDs_Bf<=TXD2_Bf;
        TX<='0';                    --start bit
        Tsend_DLN<="0000";
        TX_P_NEOSM<=ParityN(0);     --Even,Odd,Space or Mark
        Tx_B_Clr<='1';
        T_Half_f<='0';
        Tx_s<="001";
    elsif Tx_s/=0 then
        Tx_B_Clr<='0';
        T_Half_f<=not T_Half_f;
        case Tx_s is
            when "001" =>
                if T_Half_f='1' then
                    if Tsend_DLN=DLN then
                        if ParityN(2)='0' then --None Parity Bit
                            Tx_s<=StopNn;
                            TX<='1';            --Stop Bit
                        else
                            TX<=TX_P_NEOSM;     --Parity Bit
                            Tx_s<="010";
                        end if;
                    else
                        if ParityN(1)='0' then
                            TX_P_NEOSM<=TX_P_NEOSM xor TXDs_Bf(0);
                            --Even or Odd
                        end if;
                        TX<=TXDs_Bf(0);         --Send Data:Bit 0..7
                        TXDs_Bf<=TXDs_Bf(0) & TXDs_Bf(7 downto 1);
                        Tsend_DLN<=Tsend_DLN+1;
                    end if;
                end if;
            when "011" =>
                Tx_s<=StopNn;
                TX<='1';    --Stop Bit
            when others=>
                Tx_s<=Tx_s+1;
        end case;
    end if;
end if;
end process TxP;

TxBaudP:process(Clk,Reset)
variable f_Div:integer range 0 to 20832;
begin
    if Reset='0' then
        f_Div:=0;Tx_f<='0';
        BaudRate1234<="00";
    elsif rising_edge(clk) then
        if f_Div=TX_BaudRate then
```

```
            f_Div:=0;
            Tx_f<=not Tx_f;
            BaudRate1234<=BaudRate1234+1;
        else
            f_Div:=f_Div+1;
        end if;
    end if;
end process TxBaudP;

with (F_Set & BaudRate1234) select
  TX_BaudRate<= --Baud Rate Set 依 Clk=25MHz 設定
        20832   when "00000",--300:25000000/((20832+1)*4)=300.0048001
        20832   when "00001",--300
        20832   when "00010",--300
        20832   when "00011",--300
        10416   when "00100",--600
        10416   when "00101",--600
        10416   when "00110",--600
        10416   when "00111",--600
        5207    when "01000",--1200
        5207    when "01001",--1200
        5207    when "01010",--1200
        5207    when "01011",--1200
        2603    when "01100",--2400
        2603    when "01101",--2400
        2603    when "01110",--2400
        2603    when "01111",--2400
        1301    when "10000",--4800
        1301    when "10001",--4800
        1301    when "10010",--4800
        1301    when "10011",--4800
        650     when "10100",--9600
        650     when "10101",--9600
        650     when "10110",--9600
        650     when "10111",--9600
        324     when "11000",--19200
        325     when "11001",--19200 校正頻率
        324     when "11010",--19200
        325     when "11011",--19200 校正頻率
        162     when "11100",--38400
        162     when "11101",--38400
        161     when "11110",--38400 校正頻率
        162     when "11111",--38400
        0       when others;

with DL select                  --Data Length
  DLN<="0101"   when "00",      --5bit
       "0110"   when "01",      --6bit
       "0111"   when "10",      --7bit
       "1000"   when "11",      --8bit
       "0000"   when others;

with StopN select               --Stop Bit
```

```
  StopNn<="101" when "10",      --2Bit
          "110" when "11",      --1.5Bit
          "111" when others;    --1Bit

end Albert;
```

RS232_R2.vhd

```
--RS232RX
Library IEEE;
Use IEEE.std_logic_1164.all;
Use IEEE.std_logic_unsigned.all;

entity RS232_R2 is
    port(clk,Reset:in std_logic;--clk:25MHz
         DL:in std_logic_vector(1 downto 0);
         --00:5,01:6,10:7,11:8 Bit
         ParityN:in std_logic_vector(2 downto 0);
         --0xx:None,100:Even,101:Odd,110:Space,111:Mark
         StopN:in std_logic_vector(1 downto 0);
         --0x:1Bit,10:2Bit,11:1.5Bit
         F_Set:in std_logic_vector(2 downto 0);
         Status_s:out std_logic_vector(2 downto 0);
         Rx_R:in std_logic;
         RD:in std_logic;
         RxDs:out std_logic_vector(7 downto 0));
end RS232_R2;

architecture Albert of RS232_R2 is
signal StopNn:std_logic_vector(2 downto 0);
signal Rx_B_Empty,Rx_P_Error,Rx_OW,Rx_R2:std_logic;

signal RDf,Rx_f,Rx_PEOSM,R_Half_f:std_logic;
signal RxD,RxDB:std_logic_vector(7 downto 0);
signal Rsend_RDLNs,RDLN:std_logic_vector(3 downto 0);
signal Rc:std_logic_vector(2 downto 0);
signal Rx_s,Rff,BaudRate1234:std_logic_vector(1 downto 0);
signal RX_BaudRate:integer range 0 to 20832;

begin
Status_s<=Rx_B_Empty & Rx_P_Error & Rx_OW;
RDf<=clk when (Rx_s(0) = Rx_s(1)) else Rx_f;

RxP:process(RDf,Reset)
begin
if Reset='0' then
    Rx_OW<='0';
    Rx_B_Empty<='0';
    Rx_P_Error<='0';
```

```
    Rx_R2<=Rx_R;
    Rx_s<="00";
elsif falling_edge(RDf) then
    if Rx_R2/=Rx_R and Rsend_RDLNs/=RDLN then
        if Rx_R='1' then
            Rx_OW<='0';
            Rx_B_Empty<='0';
            Rx_P_Error<='0';
        end if;
        Rx_R2<=Rx_R;
    end if;
    if Rx_s=0 then
        if RD='0' then      --Start Bit
            Rx_s<="01";
            R_Half_f<='1';
            Rx_PEOSM<=ParityN(0);
        end if;
        Rsend_RDLNs<="0000";
    elsif Rx_s="11" then    --Stop Bit
        Rx_s<=not (RD & RD);
    else
        R_Half_f<=not R_Half_f;
        if R_Half_f='1' then
            if Rsend_RDLNs=RDLN then
                RxDs<=RxDB;
                Rx_B_Empty<='1';            --Rx Buffer Full
                Rx_OW<=Rx_B_Empty;          --Rx Buffer Over Write
                if ParityN(2)='1' then      --Now is Parity Bit
                    if RD/=Rx_PEOSM then
                        Rx_P_Error<='1';    --Parity Error
                    end if;
                    Rx_s<="11";
                else                        --Now is Stop Bit
                    Rx_s<="00";
                end if;
            else                            --Now is Start or Data
Bit
                RxD<=RD & RxD(7 downto 1);
                Rx_PEOSM<=Rx_PEOSM xor RD;
                Rsend_RDLNs<=Rsend_RDLNs+1;--含Start Bit
            end if;
        end if;
    end if;
end if;
end process RxP;

RxBaudP:process(clk,Rx_s)
variable F_Div:integer range 0 to 20832;
begin
    if Rx_s(0)=Rx_s(1) then
        F_Div:=0;Rx_f<='1';
        BaudRate1234<="00";
    elsif rising_edge(clk) then
```

```
        if F_Div=RX_BaudRate then
            F_Div:=0;
            Rx_f<=not Rx_f;
            BaudRate1234<=BaudRate1234+1;
        else
            F_Div:=F_Div+1;
        end if;
    end if;
end process RxBaudP;

with (F_Set & BaudRate1234) select
  RX_BaudRate<= --Baud Rate Set 依 Clk=25MHz 設定
        20832   when                                          "00000",--
300:25000000/((20832+1)*4)=300.0048001
        20832   when "00001",   --300
        20832   when "00010",   --300
        20832   when "00011",   --300
        10416   when "00100",   --600
        10416   when "00101",   --600
        10416   when "00110",   --600
        10416   when "00111",   --600
        5207    when "01000",   --1200
        5207    when "01001",   --1200
        5207    when "01010",   --1200
        5207    when "01011",   --1200
        2603    when "01100",   --2400
        2603    when "01101",   --2400
        2603    when "01110",   --2400
        2603    when "01111",   --2400
        1301    when "10000",   --4800
        1301    when "10001",   --4800
        1301    when "10010",   --4800
        1301    when "10011",   --4800
        650     when "10100",   --9600
        650     when "10101",   --9600
        650     when "10110",   --9600
        650     when "10111",   --9600
        324     when "11000",   --19200
        325     when "11001",   --19200 校正頻率
        324     when "11010",   --19200
        325     when "11011",   --19200 校正頻率
        162     when "11100",   --38400
        162     when "11101",   --38400
        161     when "11110",   --38400 校正頻率
        162     when "11111",   --38400
        0       when others;

with DL select  --Data Length 含 Start Bit
  RDLN<="0110"  when "00",  --5bit
        "0111"  when "01",  --6bit
        "1000"  when "10",  --7bit
        "1001"  when "11",  --8bit
        "0000"  when others;
```

```
with DL select  --Data Length
  RxDB<="000" & RxD(7 downto 3) when "00",  --5bit
        "00" & RxD(7 downto 2)  when "01",  --6bit
        "0" & RxD(7 downto 1)   when "10",  --7bit
        RxD                         when "11",  --8bit
        "11111111"              when others;

with StopN select
  StopNn<="101" when "10",      --2bit
           "110" when "11",     --1.5bit
           "111" when others;   --1bit

end Albert;
```

CH13_UART_1.vhd

```
--MCP3202 ch0_1 + USB UART 測試+PC+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 CH13_UART_1 is
port(gckp31,rstP99:in std_logic;        --系統頻率,系統 reset
     --RS232(UART)
     RD:in std_logic;
     TX:out std_logic;

     --MCP3202
     MCP3202_Di:out std_logic;
     MCP3202_Do:in std_logic;
     MCP3202_CLK,MCP3202_CS:buffer std_logic;
     CHs:buffer std_logic;

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

     );
end entity CH13_UART_1;

architecture Albert of CH13_UART_1 is

    --RS232_T1 & RS232_R2
    --RS232_T1
```

```
    component RS232_T1 is
    port(clk,Reset:in std_logic;--clk:25MHz
         DL:in std_logic_vector(1 downto 0);
         --00:5,01:6,10:7,11:8 Bit
         ParityN:in std_logic_vector(2 downto 0);
         --000:None,100:Even,101:Odd,110:Space,111:Mark
         StopN:in std_logic_vector(1 downto 0);
         --0x:1Bit,10:2Bit,11:1.5Bit
         F_Set:in std_logic_vector(2 downto 0);
         Status_s:out std_logic_vector(1 downto 0);
         TX_W:in std_logic;
         TXData:in std_logic_vector(7 downto 0);
         TX:out std_logic);
    end component;
    --RS232_R2
    component RS232_R2 is
    port(Clk,Reset:in std_logic;--clk:25MHz
         DL:in std_logic_vector(1 downto 0);
         --00:5,01:6,10:7,11:8 Bit
         ParityN:in std_logic_vector(2 downto 0);
         --0xx:None,100:Even,101:Odd,110:Space,111:Mark
         StopN:in std_logic_vector(1 downto 0);
         --0x:1Bit,10:2Bit,11:1.5Bit
         F_Set:in std_logic_vector(2 downto 0);
         Status_s:out std_logic_vector(2 downto 0);
         Rx_R:in std_logic;
         RD:in std_logic;
         RxDs:out std_logic_vector(7 downto 0));
    end component;

    constant DL:std_logic_vector(1 downto 0):="11";
    --00:5,01:6,10:7,11:8 Bit
    constant ParityN:std_logic_vector(2 downto 0):="000";
    --0xx:None,100:Even,101:Odd,110:Space,111:Mark
    constant StopN:std_logic_vector(1 downto 0):="00";
    --0x>1Bit,10>2Bit,11>1.5Bit
    constant F_Set:std_logic_vector(2 downto 0):="010";
    --1200 BaudRate

    signal S_RESET_T:std_logic;     --Rs232 reset 傳送
    signal TX_W:std_logic;          --寫入緩衝區
    signal Status_Ts:std_logic_vector(1 downto 0); --傳送狀態
    signal TXData:std_logic_vector(7 downto 0);--傳送資料

    signal S_RESET_R:std_logic;         --Rs232 reset 接收
    signal Rx_R:std_logic;              --讀出緩衝區
    signal Status_Rs:std_logic_vector(2 downto 0); --接收狀態
    signal RxDs:std_logic_vector(7 downto 0);  --接收資料

    signal CMDn,CMDn_R:integer range 0 to 3;   --Rs232 傳出數,接收數
    --上傳 PC 資料(4 byte)
    type pc_up_data_T is array(0 to 3) of std_logic_vector(7 downto 0);
    --命令
```

```
    signal pc_up_data:pc_up_data_T:=
        ("00000000","00000000","00000000","00000000");

    -- =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

    --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

    signal MCP3202_AD:integer range 0 to 4095;--MCP3202 AD 值

begin
```

```
U1: RS232_T1 port map(
        FD(0),S_RESET_T,DL,ParityN,StopN,F_Set,Status_Ts,
        TX_W,TXData,TX);        --RS232 傳送模組
U2: RS232_R2 port map(
        FD(0),S_RESET_R,DL,ParityN,StopN,F_Set,Status_Rs,
        Rx_R,RD,RxDs);          --RS232 接收模組
U3: 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

--上傳 PC 資料
TXData<=pc_up_data(CMDn-1);
--(上傳 ADC)
MCP3202_AD<=MCP3202_AD0 when CHs='0' else MCP3202_AD1;--通道選擇
pc_up_data(1)<=conv_std_logic_vector(MCP3202_AD/256,8);
--上傳 PC 資料 high byte
pc_up_data(0)<=conv_std_logic_vector(MCP3202_AD mod 256,8);
--上傳 PC 資料 low byte

Main:process(FD(17))
begin
    if rstP99='0' then          --系統重置
        Rx_R<='0';              --取消讀取信號
        TX_W<='0';              --取消資料載入信號
        S_RESET_T<='0';         --關閉 RS232 傳送
        S_RESET_R<='0';         --關閉 RS232 接收
        CMDn<=2;                --上傳 2byte(上傳 AD)
        CMDn_R<=1;              --接收數量(1byte)

        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 (Rx_R='1' and Status_Rs(2)='0') then --rs232 接收即時處理
        Rx_R<='0';              --即時取消讀取信號
    elsif rising_edge(FD(17)) then
        LCMP_RESET<='1';        --LCMP 啟動顯示
        S_RESET_T<='1';         --開啟 RS232 傳送
        S_RESET_R<='1';         --開啟 RS232 接收
        if CMDn>0 and S_RESET_T='1' then
```

```
            if Status_Ts(1)='0' then    --傳送緩衝區已空
                if TX_W='1' then
                    TX_W<='0';          --取消傳送資料載入時脈
                    CMDn<=CMDn-1;       --指標指向下一筆資料
                else
                    TX_W<='1';          --傳送資料載入時脈
                end if;
            end if;

        --已接收到 PC 命令
        elsif Status_Rs(2)='1' then     --已接收到 PC 命令
            Rx_R<='1';                  --讀取信號
            --PC 命令解析--
            CHs<=RxDs(0);   --通道選擇
        elsif 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<=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
                    CMDn<=2;            --上傳 2byte(上傳 AD)
                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 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;
```

CH13_UART_2.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;

entity KTM626 is
port(GCKP31,SResetP99:in std_logic;    --系統頻率,系統 reset
    s0,s1,s2,M0,M1,M2:in std_logic;    --操作指撥開關
    --UART BT
--  BT_RX:in std_logic;     --RD=BT_RX
--  BT_TX:out std_logic;    --TX=BT_TX
    --UART USB
--  USB_RX:in std_logic;
--  USB_TX:out std_logic;

    --UART BT(跳接)
    BT_RX:out std_logic;
    BT_TX:in std_logic;
    --UART USB(跳接)
    USB_RX:out std_logic;
    USB_TX:in std_logic;
    --指撥開關 DIP15(57),切換藍牙/USB
    dip15P57:in std_logic; --on(0)=USB, off(1)=BT

    --DHT11 i/o
    DHT11_D_io:inout std_logic;
    --LCD 4bit 介面
    DB_io:inout std_logic_vector(3 downto 0);
    RSo,RWo,Eo:out std_logic;
    --LED16 秀
```

```
    led16:buffer std_logic_vector(15 downto 0);
    --蜂鳴器輸出
    sound1,sound2:buffer std_logic;
    dip16P56:in std_logic;
    --串列式 LED 信號輸出
    WS2812Bout:out std_logic;
    --MG90S 伺服機輸出
    MG90S_o0:out std_logic;
    MG90S_o1:out std_logic;
    --RGB16x16 輸出
    DM13ACLKo,DM13ASDI_Ro,DM13ASDI_Go:out std_logic;
    DM13ASDI_Bo,DM13ALEo,DM13AOEo:out std_logic;
    Scan_DCBAo:buffer std_logic_vector(3 downto 0);
    --MCP3202 ADC
    MCP3202_Di:out std_logic;
    MCP3202_Do:in std_logic;
    MCP3202_CLK,MCP3202_CS:buffer std_logic;
    --timer0 數位時鐘
    PB7,PB8: in std_logic;                          --調時,調分按鈕
    scan:buffer unsigned(3 downto 0);               --掃瞄信號
    D7data:out std_logic_vector(6 downto 0);        --顯示資料
    D7xx_xx:out std_logic;                          --:閃秒
    --旋轉編碼器_選擇
    APi,BPi,PBi:in std_logic;
    --鍵盤_選擇
    keyi:in std_logic_vector(3 downto 0);           --鍵盤輸入
    keyo:buffer std_logic_vector(3 downto 0)        --鍵盤輸出
    );
end KTM626;

architecture Albert of KTM626 is
-- ================= 宣告零件 ====================
    --UART_T1 & RS232_R2
    --UART_T1--
    component RS232_T1 is
    port(clk,Reset:in std_logic;
        --clk:25MHz
        DL:in std_logic_vector(1 downto 0);
        --00:5,01:6,10:7,11:8 Bit
        ParityN:in std_logic_vector(2 downto 0);
        --000:None,100:Even,101:Odd,110:Space,111:Mark
        StopN:in std_logic_vector(1 downto 0);
        --0x:1Bit,10:2Bit,11:1.5Bit
        F_Set:in std_logic_vector(2 downto 0); --鮑率設定
        Status_s:out std_logic_vector(1 downto 0);
        TX_W:in std_logic;
        TXData:in std_logic_vector(7 downto 0);
        TX:out std_logic);
    end component RS232_T1;
    --UART_R2--
    component RS232_R2 is
    port(Clk,Reset:in std_logic;--clk:25MHz
        DL:in std_logic_vector(1 downto 0);
```

```
        --00:5,01:6,10:7,11:8 Bit
        ParityN:in std_logic_vector(2 downto 0);
        --0xx:None,100:Even,101:Odd,110:Space,111:Mark
        StopN:in std_logic_vector(1 downto 0);
        --0x:1Bit,10:2Bit,11:1.5Bit
        F_Set:in std_logic_vector(2 downto 0); --鮑率設定
        Status_s:out std_logic_vector(2 downto 0);
        Rx_R:in std_logic;
        RD:in std_logic;
        RxDs:out std_logic_vector(7 downto 0));
    end component RS232_R2;
    --宣告 UART 常數與信號--
    constant DL:std_logic_vector(1 downto 0):="11";
    --00:5,01:6,10:7,11:8 Bit
    constant ParityN:std_logic_vector(2 downto 0):="000";
    --0xx:None,100:Even,101:Odd,110:Space,111:Mark
    constant StopN:std_logic_vector(1 downto 0):="00";
    --0x>1Bit,10>2Bit,11>1.5Bit
    constant F_Set:std_logic_vector(2 downto 0):="101";
    --9600 BaudRate

    signal S_RESET_T:std_logic;     --UART 傳輸重置
    signal TX_W:std_logic;
    signal Status_Ts:std_logic_vector(1 downto 0);
    signal TXData:std_logic_vector(7 downto 0);

    signal S_RESET_R:std_logic;     --UART 接收重置
    signal Rx_R:std_logic;
    signal Status_Rs:std_logic_vector(2 downto 0);
    signal RxDs:std_logic_vector(7 downto 0);

    signal RD:std_logic;
    signal TX:std_logic;

    signal CMDn,CMDn_R:integer range 0 to 3;--UART 傳出數,接收數
    --上傳 PC 資料(4 byte)
    type pc_up_data_T is array(0 to 3) of std_logic_vector(7 downto 0);
    --命令
    signal pc_up_data:pc_up_data_T:=
        ("00000000","00000000","00000000","00000000");

    constant hTemp:integer:=28;

    --DHT11 數位溫濕度感測器--
    --Data format:
    --DHT11_DBo(std_logic_vector:8bit):由 DHT11_RDp 選取輸出項
    --RDp=5:chK_SUM
    --RDp=4 + 3 + 2 + 1 + 0
    --4:濕度(整數)+3:濕度(小數)+溫度(整數)+溫度(小數)+同位檢查
    --直接輸出濕度(DHT11_DBoH)及溫度(DHT11_DBoT):integer(0~255:8bit)
    component DHT11_driver is
        port(DHT11_CLK,DHT11_RESET:in std_logic;
            -- 781250Hz(50MHz/2^6:1.28us:FD(5))操作速率,重置
```

```
            DHT11_D_io:inout std_logic;         --DHT11 i/o
            DHT11_DBo:out std_logic_vector(7 downto 0);
            --DHT11_driver 資料輸出
            DHT11_RDp:in integer range 0 to 7; --資料讀取指標
            DHT11_tryN:in integer range 0 to 7;--錯誤後嘗試幾次
            DHT11_ok,DHT11_S:buffer std_logic;
            --DHT11_driver 完成作業旗標,錯誤信息
            DHT11_DBoH,DHT11_DBoT:out integer range 0 to 255);
            --直接輸出濕度及溫度
    end component DHT11_driver;

    signal DHT11_CLK,DHT11_RESET:std_logic;
    --DHT11_CLK:781250Hz(50MHz/2^6:1.28us:FD(5))操作速率,重置
    signal DHT11_DBo:std_logic_vector(7 downto 0);
    --DHT11_driver 資料輸出
    signal DHT11_RDp:integer range 0 to 7;      --資料讀取指標 5~0
    signal DHT11_tryN:integer range 0 to 7:=3; --錯誤後嘗試幾次
    signal DHT11_ok,DHT11_S:std_logic;
    --DHT11_driver 完成作業旗標,錯誤信息
    signal DHT11_DBoH,DHT11_DBoT:integer range 0 to 255;
    --直接輸出濕度及溫度

    --WS2812B 串列式 LED 驅動器--
    component WS2812B_Driver is
        port(WS2812BCLK,WS2812BRESET,loadck:in std_logic;
            --操作頻率,重置,載入 ck
            LEDGRBdata:in std_logic_vector(23 downto 0);
            --色彩資料
            reload,emitter,WS2812Bout:out std_logic);
            --要求載入,發射狀態,發射輸出
    end component WS2812B_Driver;

    signal WS2812BCLK,WS2812BRESET:std_logic;  --操作頻率,重置
    signal loadck,reload,emitter:std_logic;--載入 ck,要求載入,發射狀態
    signal LEDGRBdata:std_logic_vector(23 downto 0);--色彩資料

    signal FD2:std_logic_vector(3 downto 0);
    --WS2812B_Driver 除頻器
    signal SpeedS,WS2812BPCK:std_logic;
    --WS2812BP 操作頻率選擇,WS2812BP 操作頻率
    signal delay:integer range 0 to 127;        --停止時間
    signal LED_WS2812B_N:integer range 0 to 127;  --WS2812B 個數指標
    constant NLED:integer range 0 to 127:=29;
    --WS2812B 個數:30 個(0~29)
    signal LED_WS2812B_shiftN:integer range 0 to 7;
    --WS2812B 移位個數指標
    signal dir_LR:std_logic_vector(15 downto 0);   --方向控制
    type LED_T is array(0 to 7) of std_logic_vector(23 downto 0);
    --圖像格式
    --圖像
    signal LED_WS2812B_T8:LED_T:=(
                    "000000001111111100000000",
                    "111111110000000000000000",
```

```
                    "00000000000000011111111",
                    "00000000000000000000000",
                    "11111111111111100000000",
                    "00000000111111111111111",
                    "11111111000000011111111",
                    "11111111111111111111111");

    --MG90S 伺服機驅動器--
    component MG90S_Driver is
    port(MG90S_CLK,MG90S_RESET:in std_logic;
        --MG90S_Driver 驅動 clk(6.25MHz),reset 信號
        MG90S_dir0:in std_logic;            --轉動方向 0
        MG90S_deg0:in integer range 0 to 90;--轉動角度 0
        MG90S_o0:out std_logic;             --Driver 輸出 0
        MG90S_dir1:in std_logic;            --轉動方向 1
        MG90S_deg1:in integer range 0 to 90;--轉動角度 1
        MG90S_o1:out std_logic);            --Driver 輸出 1
    end component MG90S_Driver;
    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

    --RGB16x16 彩色看板--
    component RGB16x16_EP3C16Q240C8 is
    port(gckp31,RGB16x16Reset:in std_logic;    --系統頻率,系統 reset
        --DM13A
        DM13ACLKo:out std_logic;
        DM13ASDI_Ro,DM13ASDI_Go,DM13ASDI_Bo:out std_logic;
        DM13ALEo,DM13AOEo:out std_logic;
        --Scan
        Scan_DCBAo:buffer std_logic_vector(3 downto 0) );
    end component RGB16x16_EP3C16Q240C8;

    -- 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 MCP3202_Driver;

    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

    --timer 數位時鐘--
    component timer0 is
    port(GCKP31,SResetP99,p20s1,p21s2: in std_logic;
        scan:buffer unsigned(3 downto 0);  --掃瞄信號
        D7data:out std_logic_vector(6 downto 0);--顯示資料
        D7xx_xx:out std_logic  );          --:
    end component timer0;

    --ROTATE_ENCODER-旋轉編碼器--
    component ROTATE_ENCODER_EP3C16Q240C8 is
    port(gckp31,ROTATEreset:in std_logic;  --系統頻率,系統 reset
        APi,BPi,PBi:in std_logic;
        rsw:buffer std_logic_vector(2 downto 0));--3 位元計數器
    end component ROTATE_ENCODER_EP3C16Q240C8;
    signal ROTATEreset:std_logic;              --重置

    --4x4 鍵盤--
    component KEYboard_EP3C16Q240C8 is
    port(gckp31,KEYboardreset:in std_logic;        --系統頻率,系統 reset
        keyi:in std_logic_vector(3 downto 0);  --鍵盤輸入
        keyo:buffer std_logic_vector(3 downto 0);  --鍵盤輸出
        ksw:out std_logic_vector(2 downto 0)); --0~7 顯示
    end component KEYboard_EP3C16Q240C8;
    signal KEYboardreset:std_logic;                --重置

    --中文 LCM 4bit 驅動器(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 LCM_4bit_driver;

    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"15",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!
            X"4B",X"54",X"4D",X"36",X"32",X"36",   --KTM626
            X"B9",X"C5",X"A6",X"7E",X"B5",X"D8",   --嘉年華
            X"20");--空白

    --LCM=1:第一列顯示區 LEDx16 跑馬燈秀
    constant 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"4C",X"45",X"44",X"78",X"31",X"36",X"B6",
            X"5D",X"B0",X"A8",X"BF",X"4F",X"A8",X"71",
            X"20",X"20",X"20",X"20");--LEDx16 跑馬燈秀

    --LCM=2:第一列顯示區 音樂 IC 及蜂鳴器測試
    constant 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"AD",X"B5",X"BC",X"D6",X"49",X"43",X"A4",
            X"CE",X"B8",X"C1", --音樂 IC 及蜂鳴器測試
            X"BB",X"EF",X"BE",X"B9",X"B4",X"FA",X"B8",
            X"D5");--音樂 IC 及蜂鳴器測試

    --LCM=3:第一列顯示區 DHT11 溫濕度測試
    signal LCM_3: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"44",X"48",X"54",X"31",X"31",X"20",X"B7",
            X"C5",X"C0",X"E3", --DHT11 溫濕度測試
            X"AB",X"D7",X"B4",X"FA",X"B8",X"D5",X"20",
            X"20");--DHT11 溫濕度測試

    --LCM=32:第二列顯示區 溫度  ℃濕度  %RH
    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"B7",X"C5",X"AB",X"D7",X"20",X"20",X"A2",
            X"4A",X"C0",X"E3",--溫度  ℃濕度  %RH
```

```
            X"AB",X"D7",X"20",X"20",X"25",X"52",X"48",
            X"20");--溫度  ℃濕度  %RH

    --LCM=4:第一列顯示區 WS2812B RGB 測試
    constant LCM_4: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"57",X"53",X"32",X"38",X"31",X"32",X"42",
            X"20",X"52",X"47", --WS2812B RGB 測試
            X"42",X"20",X"B4",X"FA",X"B8",X"D5",X"20",
            X"20");--WS2812B RGB 測試

    --LCM=5:第一列顯示區 機械臂測試
    constant 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"BE",X"F7",X"B1",X"F1",X"C1",X"75",X"B4",
            X"FA",X"B8",X"D5",  --機械臂測試
            X"20",X"20",X"20",X"20",X"20",X"20",X"20",
            X"20");--機械臂測試

    --LCM=6:第一列顯示區 RGB16x16 秀
    constant LCM_6: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"52",X"47",X"42",X"31",X"36",X"78",X"31",
            X"36",X"A8",X"71",  --RGB16x16 秀
            X"20",X"20",X"20",X"20",X"20",X"20",X"20",
            X"20");--RGB16x16 秀

    --LCM=7:第一列顯示區 LM35 溫度測試
    constant LCM_7: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"B7",X"C5",X"AB",
            X"D7",X"B4",X"FA",  --LM35 溫度測試
            X"B8",X"D5",X"20",X"20",X"20",X"20",X"20",
            X"20");--LM35 溫度測試

    --LCM=72:第二列顯示區 溫度 xxx.x℃
    signal LCM_72: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"B7",X"C5",X"AB",X"D7",X"20",X"20",
            X"20",X"2E",X"20",X"20",--溫度 xxx.x℃
            X"A2",X"4A",X"20",X"20",X"20",X"20",
            X"20",X"20");--溫度 xxx.x℃
```

```
    signal LCM_com_data,LCM_com_data2:LCM_T;
    signal LCM_INI:integer range 0 to 31;
    signal LCMP_RESET,LN,LCMPok:std_logic;
    signal LCM,LCMx:integer range 0 to 7;

--宣告其他信號--
    signal FD:std_logic_vector(30 downto 0);   --除頻器
    signal times:integer range 0 to 2047;      --計時器
    signal S0S,S1S,S2S,M0S,M1S,M2S:std_logic_vector(2 downto 0);
    --S0,S1,S2,M0,M1,M2 防彈跳
    signal MMx,MM,PCswx,rsw,ksw:std_logic_vector(2 downto 0);
    signal LED_LR_dir,SW_CLK,sound1on:std_logic;
    signal LCD_refresh,WS2812BPReset:std_logic;
    signal MG90S_sch,MG90S_s,RGB16x16Reset:std_logic;
    signal autoMM:std_logic_vector(2 downto 0);
    signal lm35T:integer range 0 to 1550;
begin
--連接 RS232 零件--
--RS232 傳送模組
U1: RS232_T1 port map(
        FD(0),S_RESET_T,DL,ParityN,StopN,F_Set,Status_Ts,
        TX_W,TXData,TX);
--RS232 接收模組
U2: RS232_R2 port map(
        FD(0),S_RESET_R,DL,ParityN,StopN,F_Set,Status_Rs,
        Rx_R,RD,RxDs);

--連接 DHT11 零件--
DHT11_CLK<=FD(5);
--DHT11_CLK:781250Hz(50MHz/2^6:1.28us:FD(5))操作速率
U3: DHT11_driver port map(
        DHT11_CLK,DHT11_RESET,
        --781250Hz(50MHz/2^6:1.28us:FD(5))操作速率,重置
        DHT11_D_io,     --DHT11 i/o
        DHT11_DBo,      --DHT11_driver 資料輸出
        DHT11_RDp,      --資料讀取指標
        DHT11_tryN,     --錯誤後嘗試幾次
        DHT11_ok,DHT11_S,DHT11_DBoH,DHT11_DBoT);
        --DHT11_driver 完成作業旗標,錯誤信息,直接輸出濕度及溫度
--連接 LCM 零件--
LCMset: LCM_4bit_driver port map(
            FD(7),LCM_RESET,RS,RW,DBi,DBo,DB_io,
            RSo,RWo,Eo,LCMok,LCM_S);

--連接 WS2812B 零件--
WS2812BN: WS2812B_Driver port map(
            WS2812BCLK,WS2812BRESET,loadck,LEDGRBdata,
            reload,emitter,WS2812Bout);
    WS2812BRESET<=SResetP99;    --系統 reset

--連接 MG90S 零件--
MG90S: MG90S_Driver port map(
            FD(2),MG90S_RESET,
```

```
            --MG90S_Driver 驅動 clk(6.25MHz),reset 信號
            MG90S_dir0,     --轉動方向 0
            MG90S_deg0,     --轉動角度 0
            MG90S_o0,       --Driver 輸出 0
            MG90S_dir1,     --轉動方向 1
            MG90S_deg1,     --轉動角度 1
            MG90S_o1);      --Driver 輸出 1

--連接 RGB16x16 零件--
RGB16x16:RGB16x16_EP3C16Q240C8 port map(
            gckP31,RGB16x16Reset,   --系統頻率,RGB16x16Reset
            --DM13A
            DM13ACLKo,
            DM13ASDI_Ro,DM13ASDI_Go,DM13ASDI_Bo,
            DM13ALEo,DM13AOEo,
            --Scan
            Scan_DCBAo);

--連接 MCP3202 零件--
U4: 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

--連接 時鐘 零件--
U5:timer0 port map(
            GCKP31,SResetP99,PB7,PB8,
            scan, D7data,       --掃瞄信號、顯示資料
            D7xx_xx);           --:閃秒

--連接 旋轉編碼器 零件--
U6:ROTATE_ENCODER_EP3C16Q240C8 port map(
            gckp31,ROTATEreset,     --系統頻率,系統 reset
            APi,BPi,PBi,            rsw );

--連接 4x4 鍵盤 零件--
U7:KEYboard_EP3C16Q240C8 port map(
            gckp31,KEYboardreset,   --系統頻率,系統 reset
            keyi, keyo,             --鍵盤輸入、鍵盤輸出
            ksw);                   --0~7 顯示

--透過 uart 上傳溫度、濕度資料
TXData<=pc_up_data(CMDn-1);
pc_up_data(1)<=  conv_std_logic_vector(DHT11_DBoT,8) when MM="011" else
                 conv_std_logic_vector(MCP3202_AD1/256,8);
                 --上傳 PC 資料 DHT11 溫度 or LM35 ADC
pc_up_data(0)<=  conv_std_logic_vector(DHT11_DBoH,8) when MM="011" else
```

```
                conv_std_logic_vector(MCP3202_AD1 mod 256,8);
                --上傳 PC 資料 DHT11 濕度 or LM35 ADC

UART_command_Main:process(FD(17))
begin
    if SResetP99='0' then   --系統重置
        Rx_R<='0';          --取消讀取信號
        TX_W<='0';          --取消資料載入信號
        S_RESET_T<='0';     --關閉 UART 傳送
        S_RESET_R<='0';     --關閉 UART 接收
        CMDn<=0;            --上傳 0byte
        CMDn_R<=1;          --接收數量(1byte)
        PCswx<="000";
    elsif (Rx_R='1' and Status_Rs(2)='0') then --UART 接收即時處理
        Rx_R<='0';                  --即時取消讀取信號
    elsif rising_edge(FD(17)) then
        S_RESET_T<='1';             --開啟 UART 傳送
        S_RESET_R<='1';             --開啟 UART 接收
        if CMDn>0 and S_RESET_T='1' then   --上傳
            if Status_Ts(1)='0' then--傳送緩衝區已空
                if TX_W='1' then
                    TX_W<='0';      --取消傳送資料載入時脈
                    CMDn<=CMDn-1;   --指標指向下一筆資料
                else
                    TX_W<='1';      --傳送資料載入時脈
                end if;
            end if;
        ------------------------
        --已接收到 UART 命令
        elsif Status_Rs(2)='1' then--已接收到 UART 命令
            Rx_R<='1';              --讀取信號
            --PC 命令解析-------------------
            PCswx<=RxDs(2 downto 0);--接收 UART 命令
        end if;

        if  (MM="011" and LCD_refresh='1' and DHT11_ok='1') or
            (MM="111" and LCD_refresh='1' and MCP3202_ok='1')
        then CMDn<=2;
            --上傳 2byte(上傳 DHT11 溫濕度)
        end if;
    end if;
end process UART_command_Main;

--on(0)=USB, off(1)=BT(跳接)
RD <= USB_TX    when dip15P57='0' else BT_TX;
USB_RX <= TX    when dip15P57='0' else 'Z';
BT_RX <= TX     when dip15P57='1' else 'Z';

--功能自動展示切換--
autoswitch:process(FD(30))
begin
    if S0S(2)='0' then
        autoMM<="000";      --從第 1 個功能開始
```

```
    elsif rising_edge(FD(30)) then
        autoMM<=autoMM+1;   --下一個功能
    end if;
end process autoswitch;

MMx<=autoMM when S0S(2)='1' else
M2S(2)& M1S(2)& M0S(2) when S1S(2)='0' and S2S(2)='0' else
    rsw                    when S1S(2)='0' and S2S(2)='1' else
    ksw                when S1S(2)='1' and S2S(2)='0' else
    PCswx;--執行命令來源:指撥開關或 PC

KTM626_Main:process(FD(17))
begin
    if SResetP99='0' then   --系統重置
        MM<=not MMx;        --虛擬指撥開關不等於實體指撥開關狀態
        led16<=(others=>'1');--關閉 16 個 LED
        LCMP_RESET<='0';
        LCM<=0;
        sound1on<='0';
        sound2<='0';
        DHT11_RESET<='0';
        WS2812BPReset<='0';
        MG90S_RESET<='0';
        RGB16x16Reset<='0';
        MCP3202_RESET<='0';
        ROTATEreset<='0';
        KEYboardreset<='0';
    elsif rising_edge(FD(17)) then
        LCMP_RESET<='1';
        ROTATEreset<='1';
        KEYboardreset<='1';
        if LCMPok='1' then
            if MM/=MMx then -- 切換展示模式
            --實體指撥開關不等於虛擬指撥開關狀態或測試程序停止(偵測 S1)
                MM<=MMx;
                --虛擬指撥開關等於實體指撥開關狀態
                DHT11_RESET<='0';       --DHT11_driver 控制旗標
                led16<=(others=>'1');   --關閉 16 個 LED
                sound1on<='0';          --關閉蜂鳴器 1
                sound2<='0';            --關閉蜂鳴器 2(音樂 IC)
                WS2812BPReset<='1';     --串列式 LED 控制旗標
                MG90S_RESET<='0';       --伺服機控制旗標
                RGB16x16Reset<='0';     --RGB 看板控制旗標
                MCP3202_RESET<='0';     --ADC 控制旗標
                case MMx is             --根據指撥開關狀態
                    when "001" =>       --001:LED16 秀
                        LED_LR_dir<='0';--設定 LED 方向
                        led16<=(others=>'0');--關閉 16 個 LED
                        times<=10;      --設定執行 LED16 秀次數
                        LCM<=1;         --LCD 顯示
                    when "010" =>       --010:蜂鳴器輸出
                        LCM<=2;         --LCD 顯示
                        times<=200;     --設定執行次數
```

```
                    when "011" =>       --011:DHT11 溫濕度測試
                        LCM<=3;         --LCD 顯示
                        times<=800;     --設定執行次數
                    when "100" =>       --100:WS2812B 串列式 LED 秀
                        WS2812BPReset<='0';
                        LCM<=4;         --LCD 顯示
                    when "101" =>       --101:MG90S 伺服機秀
                        LCM<=5;         --LCD 顯示
                        MG90S_dir0<='0';--設定第一台伺服機之轉動方向 0
                        MG90S_deg0<=0;  --設定第一台伺服機之轉動角度 0
                        MG90S_dir1<='0';--設定第二台伺服機之轉動方向 1
                        MG90S_deg1<=0;  --設定第二台伺服機之轉動角度 1
                        times<=10;      --設定執行次數
                        MG90S_s<='0';   --設定第一台伺服機開始執行
                        MG90S_sch<='0';--設定執行正轉
                    when "110" =>       --110: RGB16x16 彩色看板秀
                        LCM<=6;         --LCD 顯示
                    when "111" =>       --111:LM35 類比溫度測試
                        LCM<=7;         --LCD 顯示
                        times<=500;     --設定執行次數
                    when others =>  --000:關閉
                        LED_LR_dir<='0';--設定 LED 方向
                        led16<=(others=>'0');--關閉 16 個 LED
                        times<=10;      --設定執行 LED16 秀次數

                        WS2812BPReset<='0';

                        MG90S_dir0<='0';--設定第一台伺服機之轉動方向 0
                        MG90S_deg0<=0;  --設定第一台伺服機之轉動角度 0
                        MG90S_dir1<='0';--設定第二台伺服機之轉動方向 1
                        MG90S_deg1<=0;  --設定第二台伺服機之轉動角度 1
                        times<=10;      --設定執行次數
                        MG90S_s<='0';   --設定第一台伺服機開始執行
                        MG90S_sch<='0';--設定執行正轉

                        LCM<=0;         --LCD 顯示

                end case;
            else        -- 執行展示模式
                times<=times-1;
                case MMx is

                    --000--
                    --  MG90S_dir0=0、  MG90S_deg0=0
                    --  MG90S_dir1=0、  MG90S_deg1=0
                    --  times=10、MG90S_s=0(第一台)、MG90S_sch=0(正轉)
                    --  LCM<=5
                    when "000" =>   --101:MG90S 伺服機測試

                        if times=0 then
                            times<=10;
                            if LED_LR_dir='1' then
                                led16<=led16(14 downto 0)&not led16(15);
```

```
                --16bit 左旋:強生技法
                LED_LR_dir<=led16(15) or not led16(14);
            else
                led16<=not led16(0)&led16(15 downto 1);
                --16bit 右旋:強生技法
                LED_LR_dir<=led16(1) and not led16(0);
            end if;
        end if;

        if dip16P56='0' then
            sound2<='1'; ---音樂 IC 連續
        else
            sound2<='0'; ---音樂 IC 不連續
        end if;

        RGB16x16Reset<='1'; --重啟看板

        WS2812BPReset<='1';

        MG90S_RESET<='1';
        if times=0 then
            times<=3;
            if MG90S_s='0' then
            --操作第一台伺服機
                if MG90S_sch='0' then
                --正轉
                    MG90S_deg0<=MG90S_deg0+1;
                    if MG90S_deg0=90 then
                        MG90S_deg0<=89;
                        MG90S_sch<='1';
                    end if;
                else
                --反轉
                    MG90S_deg0<=MG90S_deg0-1;
                    if MG90S_deg0=0 then
                        MG90S_deg0<=0;
                        MG90S_dir0<=not MG90S_dir0;
                        MG90S_sch<='0';
                        MG90S_s<=MG90S_dir0;
                    end if;
                end if;
            else
            --操作第二台伺服機
                if MG90S_sch='0' then
                --正轉
                    MG90S_deg1<=MG90S_deg1+1;
                    if MG90S_deg1=90 then
                        MG90S_deg1<=89;
                        MG90S_sch<='1';
                    end if;
                else
                --反轉
                    MG90S_deg1<=MG90S_deg1-1;
```

```
                                if MG90S_deg1=0 then
                                    MG90S_deg1<=0;
                                    MG90S_dir1<=not MG90S_dir1;
                                    MG90S_sch<='0';
                                    MG90S_s<=not MG90S_dir1;
                                end if;
                            end if;
                        end if;
                    end if;

                --001--
                --  LED_LR_dir=0、led16=0、times=10
                --  LCM=1
                when "001" =>   --LED16  --來回:強生技法
                    if times=0 then
                        times<=10;
                        if LED_LR_dir='1' then
                            led16<=led16(14 downto 0)&not led16(15);
                            --16bit 左旋:強生技法
                            LED_LR_dir<=led16(15) or not led16(14);
                        else
                            led16<=not led16(0)&led16(15 downto 1);
                            --16bit 右旋:強生技法
                            LED_LR_dir<=led16(1) and not led16(0);
                        end if;
                    end if;

                --010--
                --times=200、LCM=2
                when "010" =>   --蜂鳴器輸出
                    sound2<='1'; ---音樂 IC

                --011--
                --times=800、LCM=3
                when "011" =>   --DHT11 溫濕度測試
                    if dip16P56='0' then
                        sound2<='1'; ---音樂 IC 連續
                    else
                        sound2<='0'; ---音樂 IC 不連續
                    end if;

                    if DHT11_RESET='0' then--DHT11_driver 尚未啟動
                        DHT11_RESET<='1';    --DHT11 資料讀取
                        LCD_refresh<='1';    --更新 LCD 旗標設定為 1
                    elsif DHT11_ok='1' then  --DHT11 讀取結束
                        if LCD_refresh='1' then--更新 LCD 上的溫濕度
                            LCMP_RESET<='0'; --重啟 LCD
                            LCD_refresh<='0';
                            --更新 LCD 旗標設定為 0
                            times<=800;
                        elsif times=0 then
                            DHT11_RESET<='0';
                            --DHT11 準備重新讀取資料
```

```
                    elsif DHT11_S='1' then
                    --資料讀取失敗
                        null;           --什麼都別做(等待)
                    elsif DHT11_DBoT>hTemp then
                    --溫度超過 hTemp 度
                        sound1on<='1';  --嗶一聲
                    else
                        sound1on<='0';
                    end if;
                end if;

            --100--
            --WS2812BPReset=0
            --LCM=4
            when "100"  =>  --WS2812B 串列式 LED 秀
                WS2812BPReset<='1';

                if dip16P56='0' then
                    sound2<='1'; ---音樂 IC 連續
                else
                    sound2<='0'; ---音樂 IC 不連續
                end if;

            --101--
            --MG90S_dir0=0、    MG90S_deg0=0
            --MG90S_dir1=0、    MG90S_deg1=0
            --times=10、MG90S_s=0(第一台)、MG90S_sch=0(正轉)
            --LCM<=5
            when "101" =>   --101:MG90S 伺服機測試
                MG90S_RESET<='1';
                if times=0 then
                    times<=3;
                    if MG90S_s='0' then
                    --操作第一台伺服機
                        if MG90S_sch='0' then
                        --正轉
                            MG90S_deg0<=MG90S_deg0+1;
                            if MG90S_deg0=90 then
                                MG90S_deg0<=89;
                                MG90S_sch<='1';
                            end if;
                        else
                        --反轉
                            MG90S_deg0<=MG90S_deg0-1;
                            if MG90S_deg0=0 then
                                MG90S_deg0<=0;
                                MG90S_dir0<=not MG90S_dir0;
                                MG90S_sch<='0';
                                MG90S_s<=MG90S_dir0;
                            end if;
                        end if;
                    else
                    --操作第二台伺服機
```

```
                                if MG90S_sch='0' then
                                --正轉
                                    MG90S_deg1<=MG90S_deg1+1;
                                    if MG90S_deg1=90 then
                                        MG90S_deg1<=89;
                                        MG90S_sch<='1';
                                    end if;
                                else
                                --反轉
                                    MG90S_deg1<=MG90S_deg1-1;
                                    if MG90S_deg1=0 then
                                        MG90S_deg1<=0;
                                        MG90S_dir1<=not MG90S_dir1;
                                        MG90S_sch<='0';
                                        MG90S_s<=not MG90S_dir1;
                                    end if;
                                end if;
                            end if;
                        end if;

                    --110--
                    --LCM=6
                    when "110" =>   --RGB16x16 test
                        RGB16x16Reset<='1'; --重啟看板

                    --111--
                    --  times=500、LCM=7
                    when "111" =>    --LM35類比溫度感測
                        if MCP3202_RESET='0' then--LM35_driver尚未啟動
                            MCP3202_RESET<='1';  --LM35資料讀取
                            LCD_refresh<='1';
                        elsif MCP3202_ok='1' then--LM35讀取結束
                            if LCD_refresh='1' then
                                LCMP_RESET<='0';
                                LCD_refresh<='0';
                                times<=500;
                            elsif times=0 then    --時間到
                                MCP3202_RESET<='0';--LM35準備重新讀取資料
                            elsif MCP3202_S='1' then--資料讀取失敗
                                null;             --什麼都不做(等待)
                            end if;
                        end if;

                        if dip16P56='0' then
                            sound2<='1';--音樂IC連續
                        else
                            sound2<='0';--音樂IC不連續
                        end if;

                    when others =>      --什麼都不做(等待)
                        null;
                end case;
            end if;
```

```
        end if;
    end if;
end process KTM626_Main;

--嗶聲--
sound1<=FD(20)and FD(16)and FD(11)and sound1on when MM="010"
            else FD(22)and FD(16) and sound1on;

--DHT11 LCM顯示
LCM_32(16)<="0011" & conv_std_logic_vector(DHT11_DBoH mod 10,4);
-- 擷取濕度之個位數(ASCII)
LCM_32(15)<="0011" & conv_std_logic_vector((DHT11_DBoH/10)mod 10,4);
-- 擷取濕度之十位數(ASCII)
LCM_32(8)<="0011" & conv_std_logic_vector(DHT11_DBoT mod 10,4);
-- 擷取溫度之個位數(ASCII)
LCM_32(7)<="0011" & conv_std_logic_vector((DHT11_DBoT/10)mod 10,4);
-- 擷取溫度之十位數(ASCII)

--LM35 LCM顯示
LM35T<=MCP3202_AD1*122/100;
--5/10mv=500/4095*1000=122*MCP3202_AD1/100 xxx.x
-- MCP3202為12bit ADC(0~4095)，電壓範圍為0~5V
--每個MCP3202刻度5/4095 V，或5/4095*1000 mV=1.22mV
-- LM35每一度改變10mV，即10m/1.22個刻度=10/1.22個刻度
--若要將MCP3202轉換後的值，還原為溫度必須除以這個值

--MCP3202轉換後的值為MCP3202_ADI
--則溫度為MCP3202_ADI/(10/1.22)或MCP3202_ADI*0.122
--若要以小數一位表示，則為MCP3202_ADI*1.22，或MCP3202_ADI*122/100

LCM_72(7)<=X"20" when LM35T<1000
            else "0011" & conv_std_logic_vector(LM35T/1000,4);
--擷取百位數(ASCII)
LCM_72(8)<=X"20" when LM35T<100
             else "0011"&conv_std_logic_vector((LM35T/100)mod 10,4);
--擷取十位數(ASCII)
LCM_72(9)<="0011" & conv_std_logic_vector((LM35T/10)mod 10,4);
--擷取個位數(ASCII)
LCM_72(10)<=X"2E";
--.小數點(ASCII)
LCM_72(11)<="0011" & conv_std_logic_vector(LM35T mod 10,4);
--擷取小數1位(ASCII)

--色彩資料--
LEDGRBdata<=LED_WS2812B_T8((LED_WS2812B_N+LED_WS2812B_shiftN) mod 8)
    when MMx="100" or MMx="000" else (others=>'0');

--WS2812BP操作頻率選擇
WS2812BPCK<=FD(8) when SpeedS='0' else FD(17);
--SpeedS=0快速(97.7KHz)、SpeedS=1慢速(191Hz)
WS2812BP:process(WS2812BPCK)
begin
    if WS2812BPReset='0' then  --重置
```

```
        LED_WS2812B_N<=0;       --從頭開始
        LED_WS2812B_shiftN<=0;  --移位 0
        dir_LR<=(others=>'0');  --15..0
        loadck<='0';
        SpeedS<='0';            --加快操作速率
    elsif rising_edge(WS2812BPCK) then
        if loadck='0' then      --等待載入
            loadck<=reload;
        elsif LED_WS2812B_N=NLED then  --NLED 為 WS2812B 之數量
            SpeedS<='1';               --放慢操作速率
            if emitter='0' then        --已停止發射
                if delay/=0 then       --點亮時間&變化速率
                    delay<=delay-1;    --時間遞減
                else
                    loadck<='0';       --reemitter
                    LED_WS2812B_N<=0;  --從頭開始
                    dir_LR<=dir_LR+1;  --方向控制
                    if dir_LR(7)='1' then
                    --方向控制每 256 個 WS2812BPCK 切換一次方向移位
                        LED_WS2812B_shiftN<=LED_WS2812B_shiftN+1;
                        --移位遞增
                    else
                        LED_WS2812B_shiftN<=LED_WS2812B_shiftN-1;
                        --移位遞減
                    end if;
                    SpeedS<='0';            --加快操作速率
                end if;
            end if;
        else
            loadck<='0';
            LED_WS2812B_N<=LED_WS2812B_N+1;--調整輸出色彩
            delay<=20;      --40;
        end if;
    end if;
end process WS2812BP;

--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--LCM 更新顯示
        LCMx<=LCM;
        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
            when 2=>
```

```
                LCM_com_data<=LCM_2;    --輸出第一列資料
            when 3=>
                LCM_com_data<=LCM_3;    --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_4;    --輸出第一列資料
            when 5=>
                LCM_com_data<=LCM_5;    --輸出第一列資料
            when 6=>
                LCM_com_data<=LCM_6;    --輸出第一列資料
            when 7=>
                LCM_com_data<=LCM_7;    --輸出第一列資料
                LCM_com_data2<=LCM_72;  --輸出第二列資料
                LN<='1';                --設定輸出 2 列
            when others =>
                LCM_com_data<=LCM_IT;   --輸出第一列資料
        end case;
        LCMPok<='0';
        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) 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
                LN<='0';
                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;

SW_CLK<=FD(19); --防彈跳操作速率
process(SW_CLK) --防彈跳
begin
```

```
    --S0 防彈跳
    if S0='0' then
        S0S<="000";
    elsif rising_edge(SW_CLK) then
        S0S<=S0S+ not S0S(2);
    end if;

    --S1 防彈跳
    if S1='0' then
        S1S<="000";
    elsif rising_edge(SW_CLK) then
        S1S<=S1S+ not S1S(2);
    end if;

    --S2 防彈跳
    if S2='0' then
        S2S<="000";
    elsif rising_edge(SW_CLK) then
        S2S<=S2S+ not S2S(2);
    end if;

    --M0 防彈跳
    if M0='0' then
        M0S<="000";
    elsif rising_edge(SW_CLK) then
        M0S<=M0S+ not M0S(2);
    end if;

    --M1 防彈跳
    if M1='0' then
        M1S<="000";
    elsif rising_edge(SW_CLK) then
        M1S<=M1S+ not M1S(2);
    end if;

    --M2 防彈跳
    if M2='0' then
        M2S<="000";
    elsif rising_edge(SW_CLK) then
        M2S<=M2S+ not M2S(2);
    end if;

end process;

--除頻器--
Freq_Div:process(GCKP31)
begin
    if SResetP99='0' then           --系統 reset
        FD<=(others=>'0');
        FD2<=(others=>'0');
        WS2812BCLK<='0';            --WS2812BN 驅動頻率
    elsif rising_edge(GCKP31) then --50MHz
        FD<=FD+1;
```

```
        if FD2=9 then                --7~12
            FD2<=(others=>'0');
            WS2812BCLK<=not WS2812BCLK;--50MHz/20=2.5MHz T.=. 0.4us
        else
            FD2<=FD2+1;
        end if;
    end if;
end process Freq_Div;

end Albert;
```