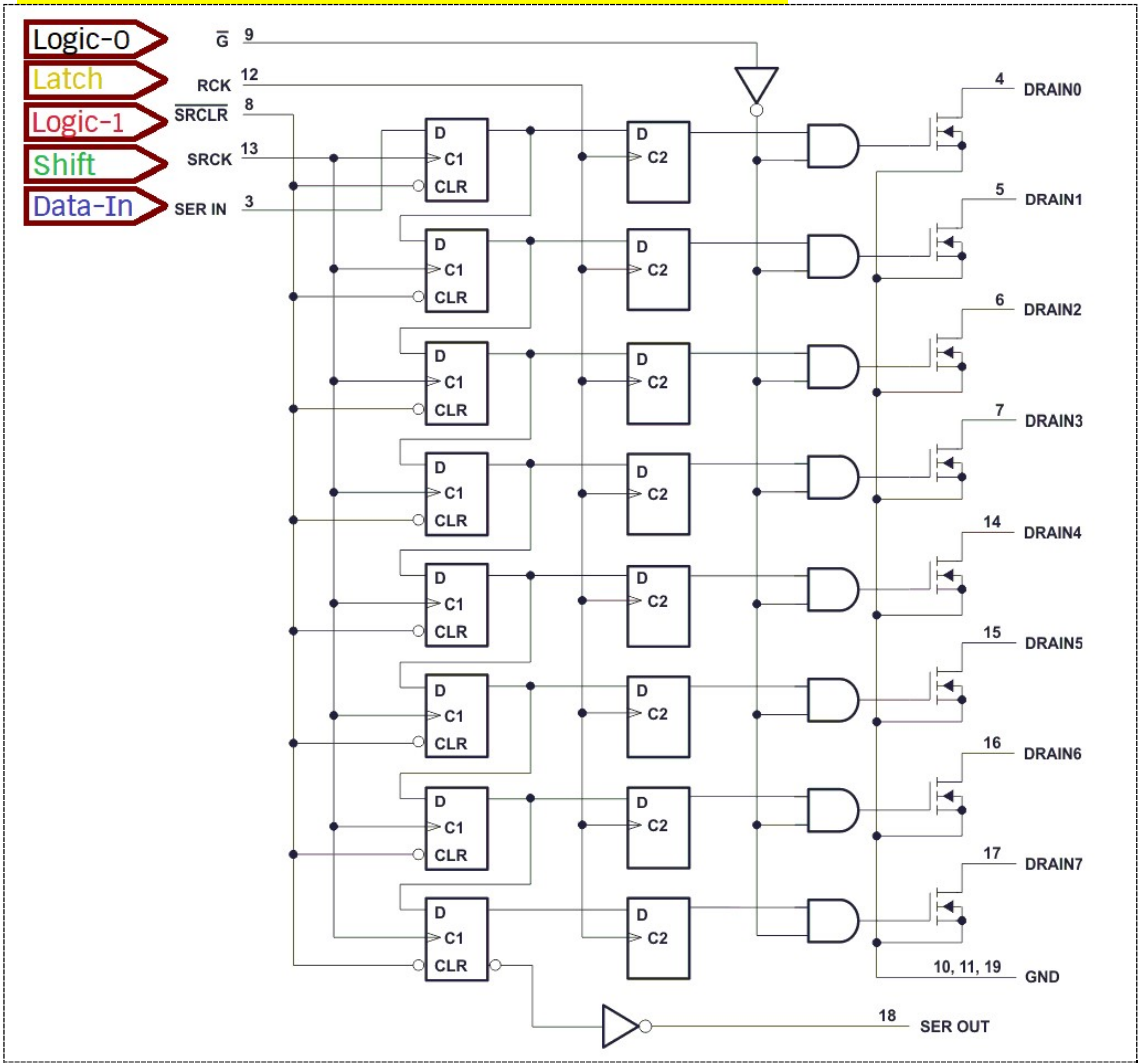
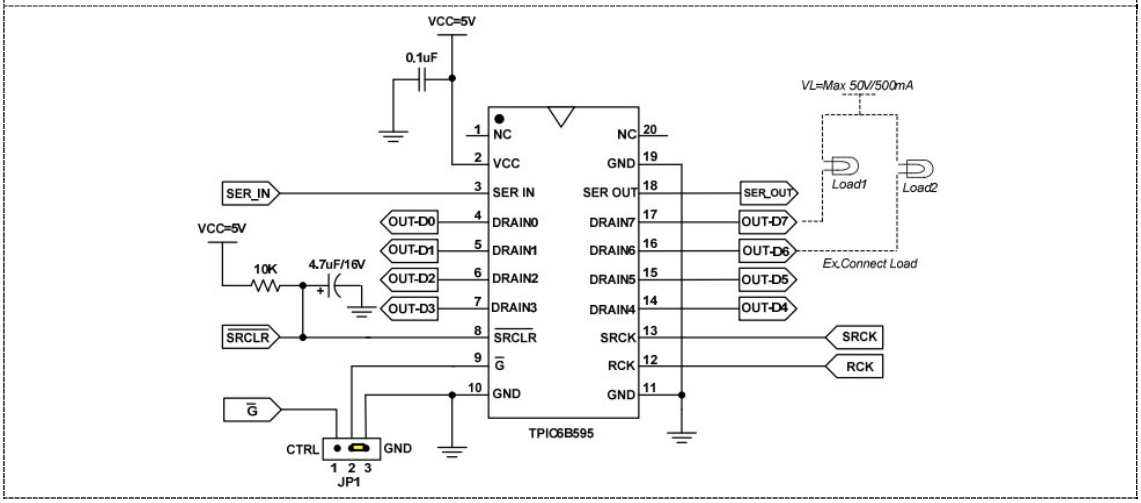
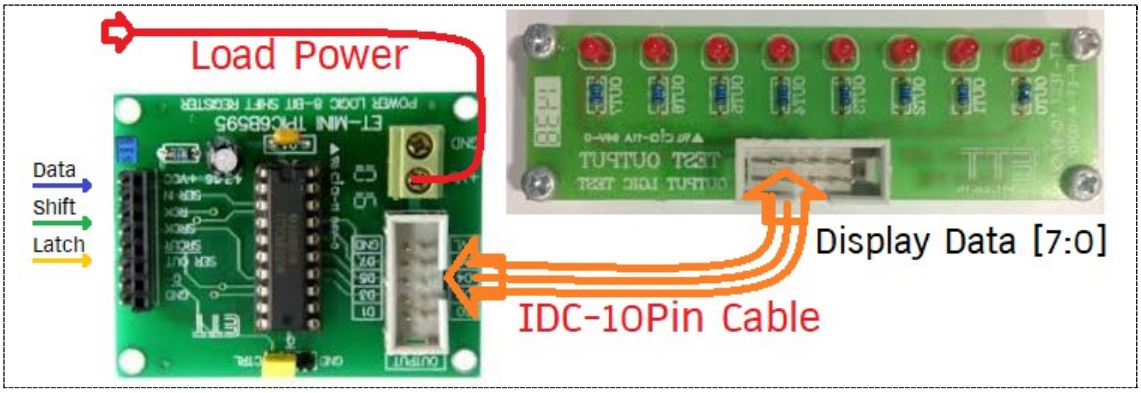
นายวิชัย ศรีสุรักษ์
B3706985

Week07_Mojo V3 - Timing Diagram Coding

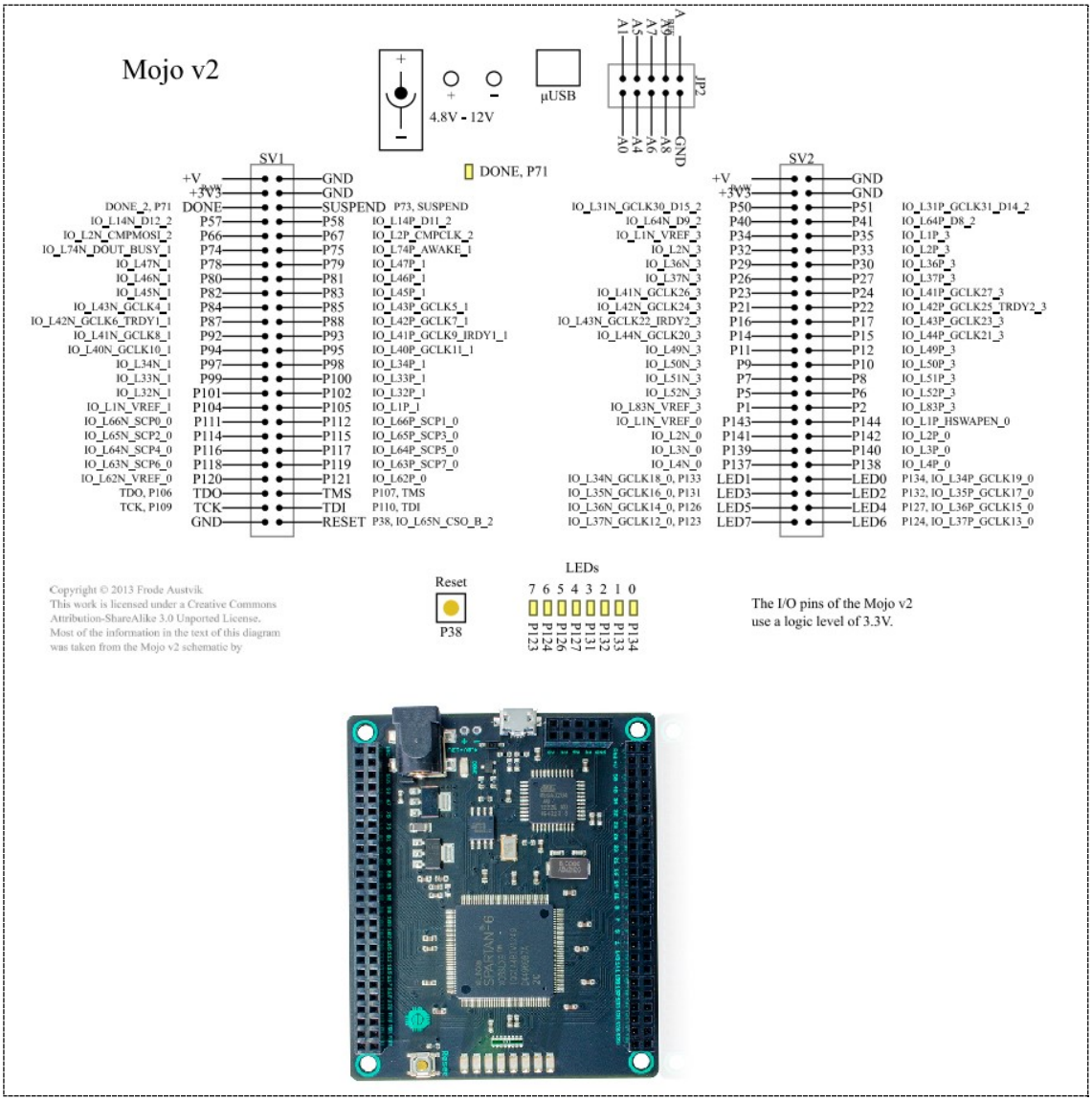
1. การใช้งาน TPIC6B595 – Power Logic 8 bit Shift register







ET-MINI TPIC6B595



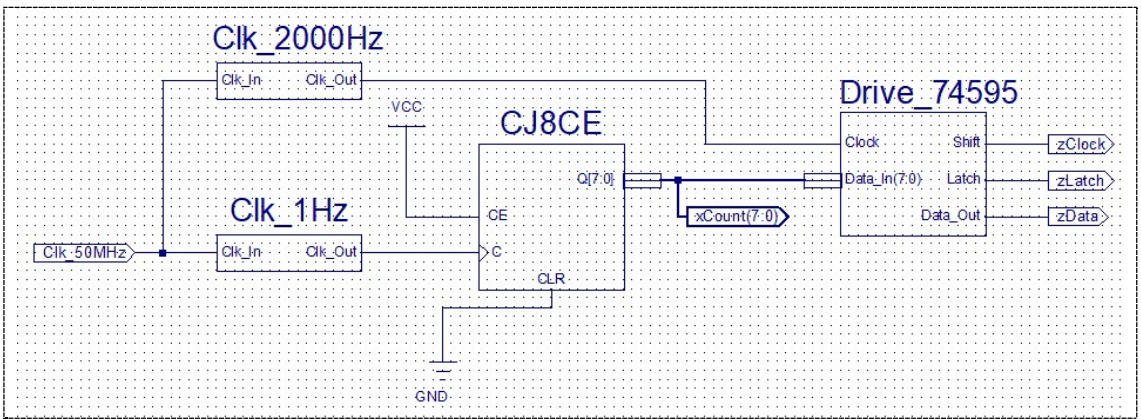
Mojo V2 Pin Out

2. การโปรแกรมใช้งาน TPIC6B595 ด้วย Arduino

```
#define zData   12
#define zShift  11
#define zKLatch 10
int XXX = 50;

void Send_TPIC6B595(int iData)
{ digitalWrite(zShift, HIGH);  digitalWrite(zKLatch, LOW);  digitalWrite(zData, (iData >> 0) & 1);   delayMicroseconds(XXX);
  digitalWrite(zShift, LOW);   digitalWrite(zKLatch, LOW);  digitalWrite(zData, (iData >> 0) & 1);   delayMicroseconds(XXX);
  digitalWrite(zShift, HIGH);  digitalWrite(zKLatch, LOW);  digitalWrite(zData, (iData >> 1) & 1);   delayMicroseconds(XXX);
  digitalWrite(zShift, LOW);   digitalWrite(zKLatch, LOW);  digitalWrite(zData, (iData >> 1) & 1);   delayMicroseconds(XXX);
  digitalWrite(zShift, HIGH);  digitalWrite(zKLatch, LOW);  digitalWrite(zData, (iData >> 2) & 1);   delayMicroseconds(XXX);
  digitalWrite(zShift, LOW);   digitalWrite(zKLatch, LOW);  digitalWrite(zData, (iData >> 2) & 1);   delayMicroseconds(XXX);
  digitalWrite(zShift, HIGH);  digitalWrite(zKLatch, LOW);  digitalWrite(zData, (iData >> 3) & 1);   delayMicroseconds(XXX);
  digitalWrite(zShift, LOW);   digitalWrite(zKLatch, LOW);  digitalWrite(zData, (iData >> 3) & 1);   delayMicroseconds(XXX);
  digitalWrite(zShift, HIGH);  digitalWrite(zKLatch, LOW);  digitalWrite(zData, (iData >> 4) & 1);   delayMicroseconds(XXX);
  digitalWrite(zShift, LOW);   digitalWrite(zKLatch, LOW);  digitalWrite(zData, (iData >> 4) & 1);   delayMicroseconds(XXX);
  digitalWrite(zShift, HIGH);  digitalWrite(zKLatch, LOW);  digitalWrite(zData, (iData >> 5) & 1);   delayMicroseconds(XXX);
  digitalWrite(zShift, LOW);   digitalWrite(zKLatch, LOW);  digitalWrite(zData, (iData >> 5) & 1);   delayMicroseconds(XXX);
  digitalWrite(zShift, HIGH);  digitalWrite(zKLatch, LOW);  digitalWrite(zData, (iData >> 6) & 1);   delayMicroseconds(XXX);
  digitalWrite(zShift, LOW);   digitalWrite(zKLatch, LOW);  digitalWrite(zData, (iData >> 6) & 1);   delayMicroseconds(XXX);
  digitalWrite(zShift, HIGH);  digitalWrite(zKLatch, LOW);  digitalWrite(zData, (iData >> 7) & 1);   delayMicroseconds(XXX);
  digitalWrite(zShift, LOW);   digitalWrite(zKLatch, LOW);  digitalWrite(zData, (iData >> 7) & 1);   delayMicroseconds(XXX);
  digitalWrite(zShift, HIGH);  digitalWrite(zKLatch, LOW);  digitalWrite(zData, (iData >> 7) & 1);   delayMicroseconds(XXX);
  digitalWrite(zShift, HIGH);  digitalWrite(zKLatch, HIGH); digitalWrite(zData, (iData >> 7) & 1);   delayMicroseconds(XXX);
  digitalWrite(zShift, HIGH);  digitalWrite(zKLatch, LOW);  digitalWrite(zData, (iData >> 7) & 1);   delayMicroseconds(XXX);
}

void setup()
{ pinMode(zData, OUTPUT);
  pinMode(zShift, OUTPUT);
  pinMode(zKLatch, OUTPUT);
}

void loop() {
  Send_TPIC6B595(0b11111111); delay(250);
  Send_TPIC6B595(0b00000000); delay(250);
  Send_TPIC6B595(0b11111111); delay(250);
  Send_TPIC6B595(0b00000000); delay(250);
  Send_TPIC6B595(0b10000000); delay(500);
  Send_TPIC6B595(0b11000000); delay(500);
  Send_TPIC6B595(0b11100000); delay(500);
  Send_TPIC6B595(0b01110000); delay(500);
  Send_TPIC6B595(0b00111000); delay(500);
  Send_TPIC6B595(0b00011100); delay(500);
  Send_TPIC6B595(0b00001110); delay(500);
  Send_TPIC6B595(0b00000111); delay(500);
  Send_TPIC6B595(0b00000011); delay(500);
  Send_TPIC6B595(0b00000001); delay(500);
}
```

3. การโปรแกรมใช้งาน TPIC6B595 ด้วย Verilog – แบบยาว

3.1/5-วงจรการทำงานหลัก



- Clock 2000Hz สำหรับระบบของไอซี 595
- Clock 1 Hz สำหรับวงจรนับเพื่อสร้างข้อมูลทดสอบ
- CJ8CE วงจรนับเพื่อสร้างข้อมูลทดสอบ
- Driver_595 วงจรขับไอซี 595

3.2/5-โมดูลสร้าง 2000Hz จาก 50MHz

```
`timescale 1ns / 1ps
module Clk_2000Hz(
   input          Clk_In,
   output         Clk_Out
   );

reg Clk_Out = 1'b0;
reg [27:0] Counter;

always@(posedge Clk_In)  begin
      Counter <= Counter + 1;
      if ( Counter == 12_500)  begin
         Counter <= 0;
         Clk_Out <= ~Clk_Out;
      end
end

endmodule
```

12500 = 30D4H (14bit)

50,000,000 / (2*12,500)
= 2000

3.3/5-โมดูลสร้าง 1Hz จาก 50MHz

```
`timescale 1ns / 1ps

module Clk_1Hz(
   input          Clk_In,
   output         Clk_Out
   );

reg Clk_Out = 1'b0;
reg [27:0] Counter;

always@(posedge Clk_In)  begin
      Counter <= Counter + 1;
      if ( Counter == 25_000_000)  begin
         Counter <= 0;
         Clk_Out <= ~Clk_Out;
      end
end

endmodule
```

25,000,000 = 17D 7840H (25bit)

50,000,000 / (2*25,000,000)
= 1

3.4/5-Pin Input-Output File

```
NET "Clk_50MHz"        LOC = P56 | IOSTANDARD = LVTTL;

NET "xCount<0>"        LOC = P134 | IOSTANDARD = LVTTL;
NET "xCount<1>"        LOC = P133 | IOSTANDARD = LVTTL;
NET "xCount<2>"        LOC = P132 | IOSTANDARD = LVTTL;
NET "xCount<3>"        LOC = P131 | IOSTANDARD = LVTTL;
NET "xCount<4>"        LOC = P127 | IOSTANDARD = LVTTL;
NET "xCount<5>"        LOC = P126 | IOSTANDARD = LVTTL;
NET "xCount<6>"        LOC = P124 | IOSTANDARD = LVTTL;
NET "xCount<7>"        LOC = P123 | IOSTANDARD = LVTTL;

NET "zData"     LOC = P51 | IOSTANDARD = LVTTL ;
NET "zClock"    LOC = P41 | IOSTANDARD = LVTTL ;
NET "zLatch"    LOC = P35 | IOSTANDARD = LVTTL ;
```

Clock 50MHz

LED Build in Monitor

Data
Shift
Latch

3.5/5-Driver_595 Verilog Module

```
`timescale 1ns / 1ps
module Drive_74595(Clock, Data_In, Shift, Latch, Data_Out);

//----------------------------CONTROL SIGNALS-----------------------------------
input   Clock;
input   [7:0] Data_In;  // 8-bit input data
output  Shift;          // Register CLK, pushes the FIFO data to the driver outputs
output  Latch;          // Positive Edge Triggered Shift Register CLK
output  Data_Out;       // The serial data output

reg [7:0] Clk_Count;
reg R_Shift, R_Latch, R_DataO;

//=======================================================
always @ ( posedge Clock )
        begin
                Clk_Count <= Clk_Count + 1;
                if( Clk_Count > 20 ) begin
                        Clk_Count       <= 0;
                end
                //-----------------------------------------------
                if(Clk_Count == 0) begin
                        R_Shift <= 0;
                        R_Latch         <= 0;
                        R_DataO         <= Data_In[0];
                end
                if(Clk_Count == 1) begin
                        R_Shift <= 1;
                        R_Latch         <= 0;
                        R_DataO         <= Data_In[0];
                end
                //-----------------------------------------------
                if(Clk_Count == 2) begin
                        R_Shift <= 0;
                        R_Latch         <= 0;
                        R_DataO         <= Data_In[1];
                end
                if(Clk_Count == 3) begin
                        R_Shift <= 1;
                        R_Latch         <= 0;
                        R_DataO         <= Data_In[1];
                end
                //-----------------------------------------------
                if(Clk_Count == 4) begin
                        R_Shift <= 0;
                        R_Latch         <= 0;
                        R_DataO         <= Data_In[2];
                end
                if(Clk_Count == 5) begin
                        R_Shift <= 1;
                        R_Latch         <= 0;
                        R_DataO         <= Data_In[2];
                end
                //-----------------------------------------------
                if(Clk_Count == 6) begin
                        R_Shift <= 0;
                        R_Latch         <= 0;
                        R_DataO         <= Data_In[3];
                end
                if(Clk_Count == 7) begin
                        R_Shift <= 1;
                        R_Latch         <= 0;
                        R_DataO         <= Data_In[3];
                end
                //-----------------------------------------------
                if(Clk_Count == 8) begin
                        R_Shift <= 0;
                        R_Latch         <= 0;
                        R_DataO         <= Data_In[4];
                end
```

```
                if(Clk_Count == 9) begin
                        R_Shift <= 1;
                        R_Latch         <= 0;
                        R_DataO         <= Data_In[4];
                end
                //-----------------------------------------------
                if(Clk_Count == 10) begin
                        R_Shift <= 0;
                        R_Latch         <= 0;
                        R_DataO         <= Data_In[5];
                end
                if(Clk_Count == 11) begin
                        R_Shift <= 1;
                        R_Latch         <= 0;
                        R_DataO         <= Data_In[5];
                end
                //-----------------------------------------------
                if(Clk_Count == 12) begin
                        R_Shift <= 0;
                        R_Latch         <= 0;
                        R_DataO         <= Data_In[6];
                end
                if(Clk_Count == 13) begin
                        R_Shift <= 1;
                        R_Latch         <= 0;
                        R_DataO         <= Data_In[6];
                end
                //-----------------------------------------------
                if(Clk_Count == 14) begin
                        R_Shift <= 0;
                        R_Latch         <= 0;
                        R_DataO         <= Data_In[7];
                end
                if(Clk_Count == 15) begin
                        R_Shift <= 1;
                        R_Latch         <= 0;
                        R_DataO         <= Data_In[7];
                end
                //-----------------------------------------------
                if(Clk_Count == 16) begin
                        R_Shift <= 0;
                        R_Latch         <= 0;
                        R_DataO         <= 0;
                end
                if(Clk_Count == 17) begin
                        R_Shift <= 0;
                        R_Latch         <= 1;
                        R_DataO         <= 0;
                end
                //-----------------------------------------------
        end

        assign Shift            = R_Shift;
        assign Latch            = R_Latch;
        assign Data_Out         = R_DataO;

endmodule
```

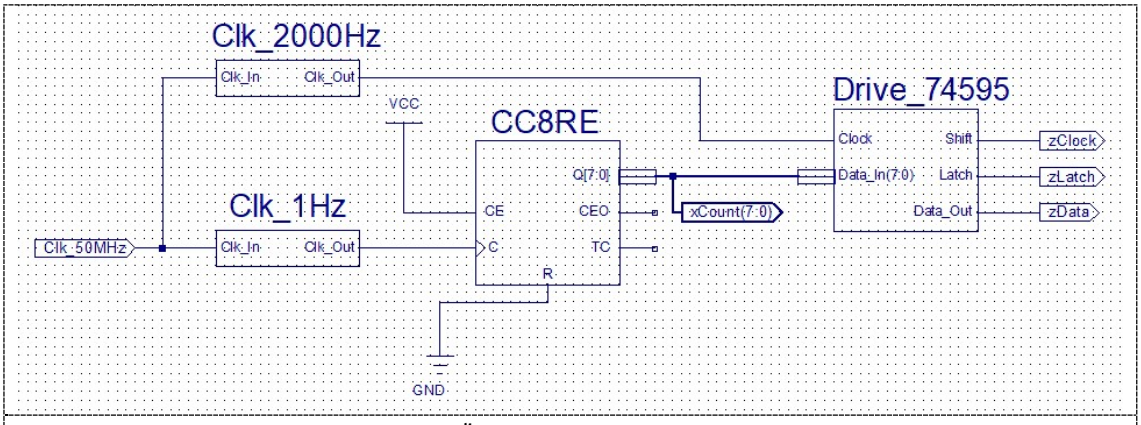
4. การโปรแกรมใช้งาน TPIC6B595 ด้วย Verilog – แบบสั้น

4.5/5-Driver_595 Verilog Module

```
`timescale 1ns / 1ps
module Drive_74595(Clock, Data_In, Shift, Latch, Data_Out);

//----------------------------CONTROL SIGNALS-----------------------------------
input    Clock;
input    [7:0] Data_In;  // 8-bit input data
output   Shift;          // Shift -> Register CLK in FIFO Buffer
output   Latch;          // Latch --> Shift Register CLK
output   Data_Out;       // The serial data output

reg [7:0] Clk_Count;     // Maximum 256 Input_Clock
reg R_Shift, R_Latch, R_DataO;
parameter nBit = 8;      // Using (2*nBit+4) Input_Clock

//=================================================
always @ ( posedge Clock ) begin
        Clk_Count <= Clk_Count + 1;
        if( Clk_Count > (2*nBit + 4)) begin
                Clk_Count       <= 0;
        end
        //--- Shift Process ---------------------------------------
        if((Clk_Count < 2*nBit) && ((Clk_Count%2) == 0)) begin
                R_Shift <= 0;
                R_Latch         <= 0;
                R_DataO         <= Data_In[Clk_Count/2];
        end
        if((Clk_Count < 2*nBit) && ((Clk_Count%2) == 1)) begin
                R_Shift <= 1;
                R_Latch         <= 0;
                R_DataO         <= Data_In[Clk_Count/2];
        end
        //--- Latch Process ---------------------------------------
        if(Clk_Count == (2*nBit)) begin
                R_Shift <= 0;
                R_Latch         <= 0;
                R_DataO         <= 0;
        end
        if(Clk_Count == (2*nBit+1)) begin
                R_Shift <= 0;
                R_Latch         <= 1;
                R_DataO         <= 0;
        end
        //-------------------------------------------------------
end

assign Shift            = R_Shift;
assign Latch            = R_Latch;
assign Data_Out         = R_DataO;

endmodule
```

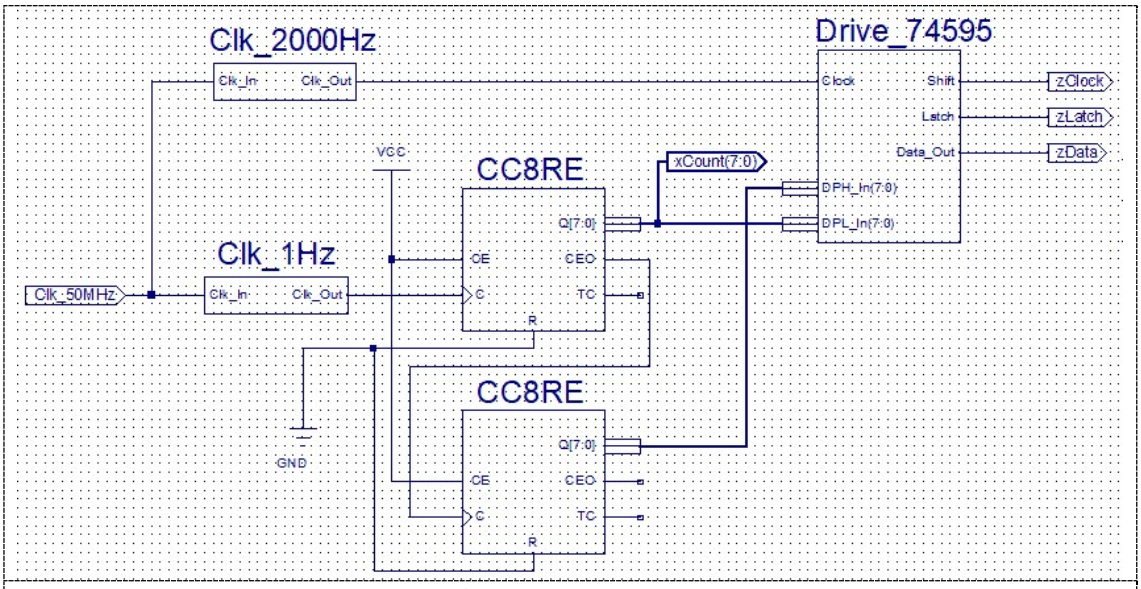
5. ปรับแก้เป็น 2 IC สำหรับ 16 bit Output

ทดสอบ 8 bit Counter



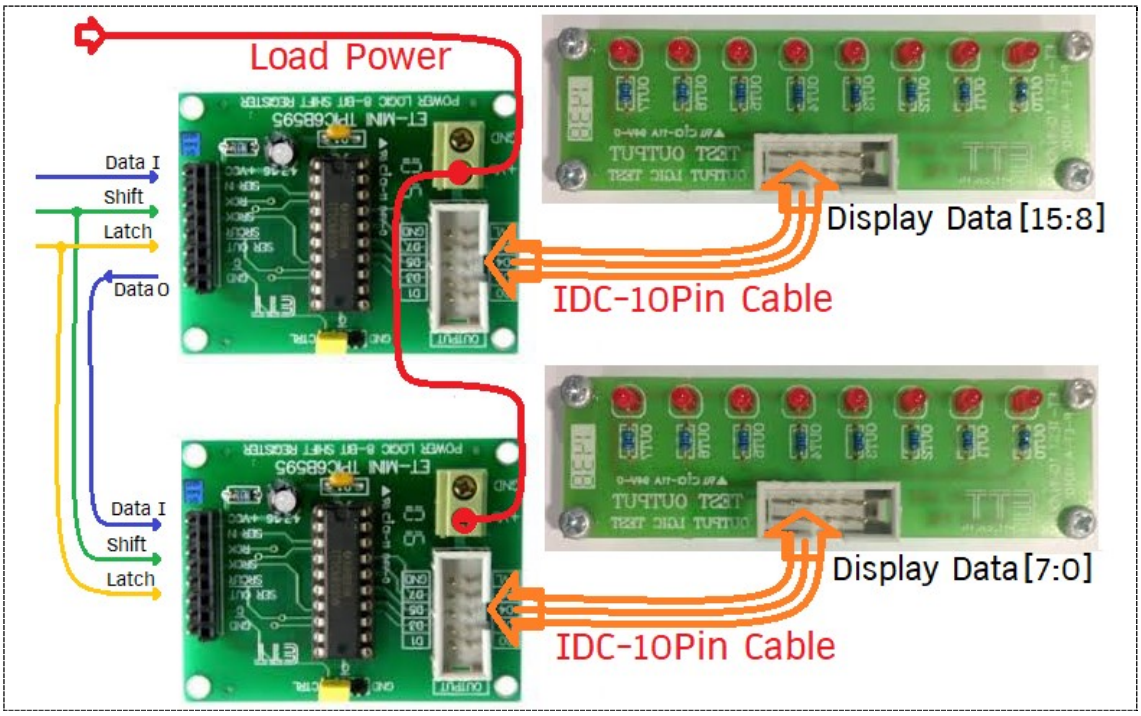
- Clock 2000Hz สำหรับระบบของไอซี 595
- Clock 1 Hz สำหรับวงจรนับเพื่อสร้างข้อมูลทดสอบ (สามารถปรับให้เร็วขึ้นก็ได้)
- CC8RE วงจรนับแบบ 8 บิตเพื่อสร้างข้อมูลทดสอบ
- Driver_595 วงจรขับไอซี 595

ทดสอบ 16 bit Counter

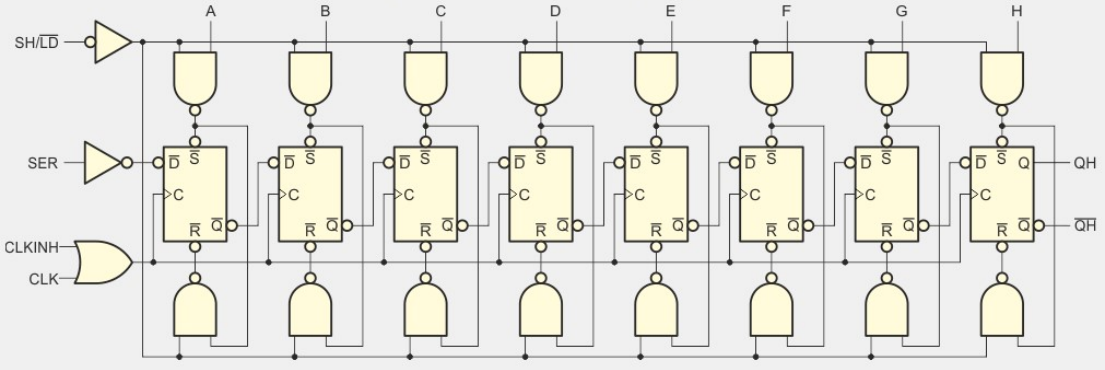


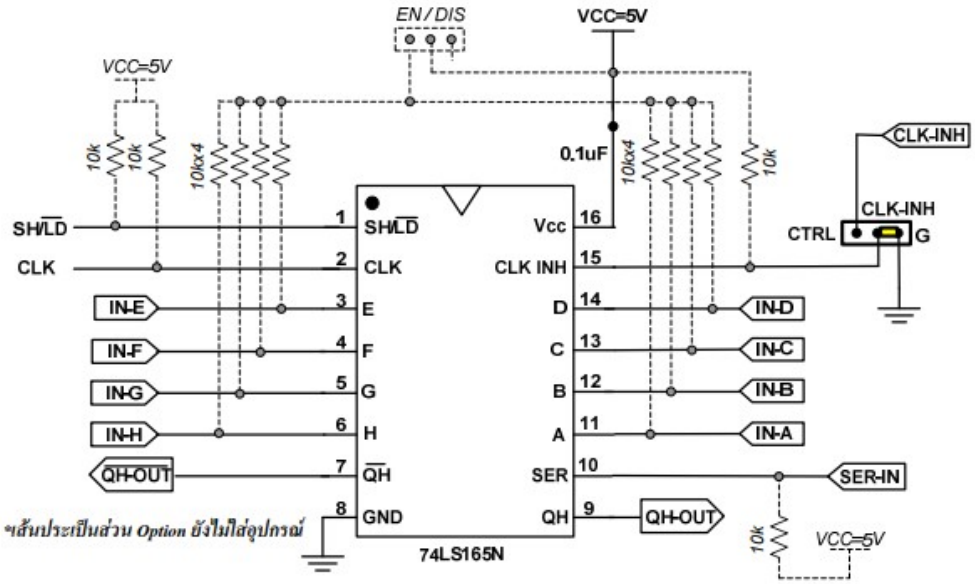
- Clock 2000Hz สำหรับระบบของไอซี 595
- Clock 1 Hz สำหรับวงจรนับเพื่อสร้างข้อมูลทดสอบ (สามารถปรับให้เร็วขึ้นก็ได้)
- CC8RE วงจรนับแบบ 8 บิตจำนวน 2 ชุด เพื่อสร้างข้อมูลทดสอบขนาด 16 บิต
- Driver_595 วงจรขับไอซี 595 ที่ปรับเป็นอินพุต 16 บิต

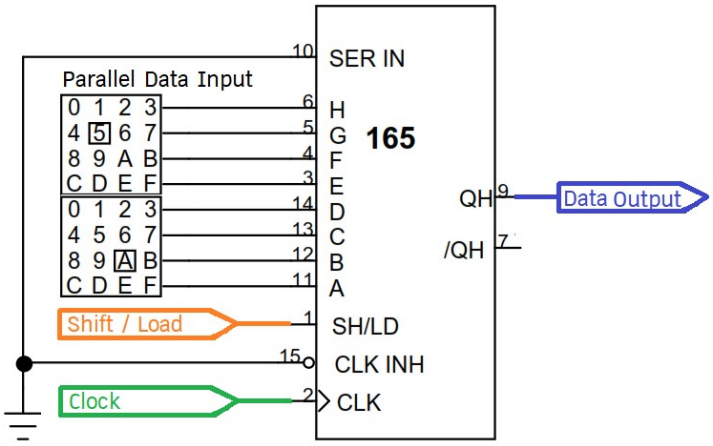
```
`timescale 1ns / 1ps
module Drive_74595(Clock, DPH_In, DPL_In, Shift, Latch, Data_Out);

//----------------------------CONTROL SIGNALS----------------------------------
input   Clock;
input   [7:0] DPH_In; // 8-bit input data H-Byte
input   [7:0] DPL_In; // 8-bit input data L-Byte
output  Shift;          // Shift -> Register CLK in FIFO Buffer
output  Latch;          // Latch --> Shift Register CLK
output  Data_Out;               // The serial data output

parameter nBit = 16;            // Using (2*nBit+4) Input_Clock
reg [7:0] Clk_Count;            // Maximum 256 Input_Clock
reg R_Shift, R_Latch, R_DataO;
reg [nBit-1:0] Data_In;

//============================================================
always @* begin
        Data_In = { DPH_In, DPL_In };
end

//============================================================
always @ ( posedge Clock ) begin
        Clk_Count <= Clk_Count + 1;
        if( Clk_Count > (2*nBit + 4)) begin
                Clk_Count <= 0;
        end
        //--- Shift Process ------------------------------------
        if((Clk_Count < 2*nBit) && ((Clk_Count%2) == 0)) begin
                R_Shift <= 0;
                R_Latch         <= 0;
                R_DataO         <= Data_In[Clk_Count/2];
        end
        if((Clk_Count < 2*nBit) && ((Clk_Count%2) == 1)) begin
                R_Shift <= 1;
                R_Latch         <= 0;
                R_DataO         <= Data_In[Clk_Count/2];
        end
        //--- Latch Process ------------------------------------
        if(Clk_Count == (2*nBit)) begin
                R_Shift <= 0;
                R_Latch         <= 0;
                R_DataO         <= 0;
        end
        if(Clk_Count == (2*nBit+1)) begin
                R_Shift <= 0;
                R_Latch         <= 1;
                R_DataO         <= 0;
        end
        //------------------------------------------------------
end

assign Shift            = R_Shift;
assign Latch            = R_Latch;
assign Data_Out         = R_DataO;

endmodule
```

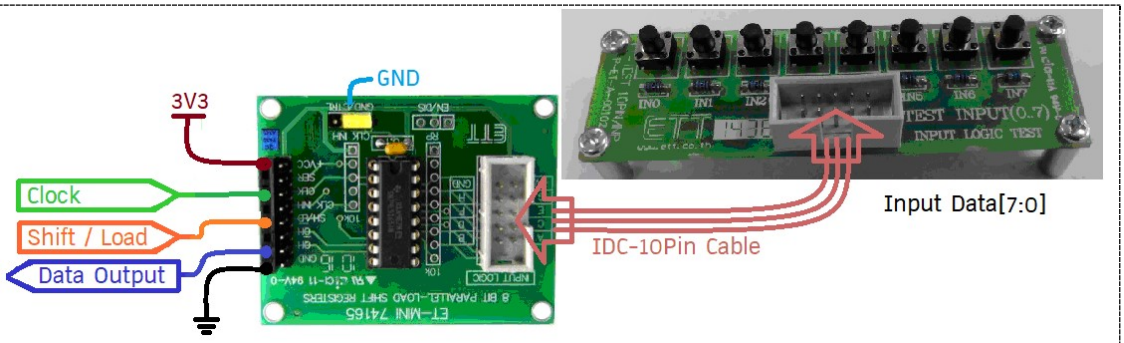

Load Power
Data I
Shift
Latch
Data O
Data I
Shift
Latch
Display Data [15:8]
IDC-10Pin Cable
Display Data [7:0]
IDC-10Pin Cable

6. การใช้งาน 74165 – Parallel Load Shift Register



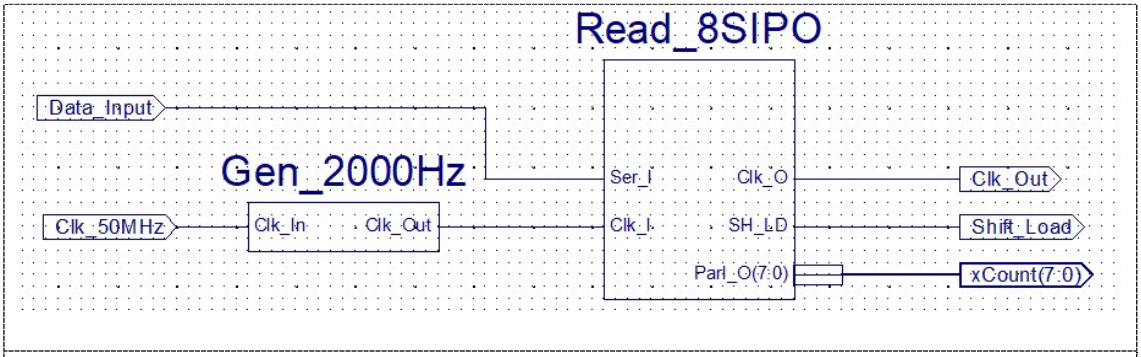






7. การโปรแกรมใช้งาน 74165 ด้วย Verilog

7.1/4-วงจรการทำงานหลัก



- Clock 2000Hz สำหรับระบบของไอซี Read_8SIPO
- Read_8SIPO วงจรรับค่าจาก 74165

7.2/4-โมดูลสร้าง 2000Hz จาก 50MHz

```
`timescale 1ns / 1ps
module Clk_2000Hz(
   input          Clk_In,
   output         Clk_Out
   );

reg Clk_Out = 1'b0;
reg [27:0] Counter;

always@(posedge Clk_In)  begin
      Counter <= Counter + 1;
      if ( Counter == 12_500)  begin
         Counter <= 0;
         Clk_Out <= ~Clk_Out;
      end
end

endmodule
```

12500 = 30D4H (14bit)

50,000,000 / (2*12,500)
= 2000

7.3/4-Pin Input-Output File

```
NET "Clk_50MHz"       LOC = P56 | IOSTANDARD = LVTTL;        Clock 50MHz

NET "xCount<0>"       LOC = P134 | IOSTANDARD = LVTTL;       LED Build in Monitor
NET "xCount<1>"       LOC = P133 | IOSTANDARD = LVTTL;
NET "xCount<2>"       LOC = P132 | IOSTANDARD = LVTTL;
NET "xCount<3>"       LOC = P131 | IOSTANDARD = LVTTL;
NET "xCount<4>"       LOC = P127 | IOSTANDARD = LVTTL;
NET "xCount<5>"       LOC = P126 | IOSTANDARD = LVTTL;
NET "xCount<6>"       LOC = P124 | IOSTANDARD = LVTTL;
NET "xCount<7>"       LOC = P123 | IOSTANDARD = LVTTL;

NET "Data_Input"      LOC = P51 | IOSTANDARD = LVTTL ;       Data_Iput
NET "Clk_Out"         LOC = P41 | IOSTANDARD = LVTTL ;       Clock to 74165
NET "Shift_Load"      LOC = P35 | IOSTANDARD = LVTTL ;       Shift or Load
```

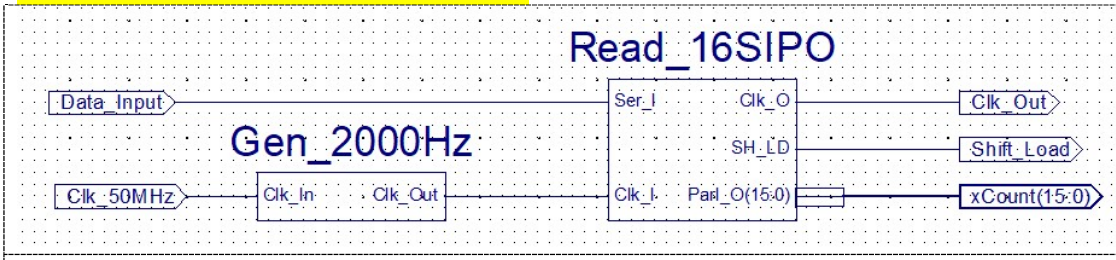
7.4/4-Read_8SIPO Verilog File

```
`timescale 1ns / 1ps

module Read_8SIPO(Ser_I, Clk_I, Clk_O, SH_LD, Parl_O);
input    Clk_I, Ser_I;
output   Clk_O, SH_LD;
output   [7:0]  Parl_O;

reg [7:0] Clk_Count;     // Maximum 256 Input_Clockreg [7:0] tmp;
parameter nBit = 8;      // Using (nBit+4) Input_Clock
reg [nBit-1:0] R_Parl_O, temp_Parl_O;
reg R_ShiftLoad;

always @(posedge Clk_I) begin
        Clk_Count <= Clk_Count + 1;
        if( Clk_Count > (nBit+4)) begin                  // Over Reset Counter
                Clk_Count       <= 0;
                R_ShiftLoad     <= 0;
        end
        if(Clk_Count==0)        R_ShiftLoad <= 0;       // Load
        if(Clk_Count==1)        R_ShiftLoad <= 1;       // no Load
        if((Clk_Count>=2)&&(Clk_Count<=(nBit+1))) begin    // Shift
                temp_Parl_O    <= temp_Parl_O << 1;
                temp_Parl_O[0] <= Ser_I;
        end
        if(Clk_Count==(nBit+2))        R_Parl_O <= temp_Parl_O;      // Save
end

assign  Parl_O  = R_Parl_O;
assign  Clk_O   = Clk_I;
assign  SH_LD   = R_ShiftLoad;

endmodule
```

7.5 คำถาม 8 บิต

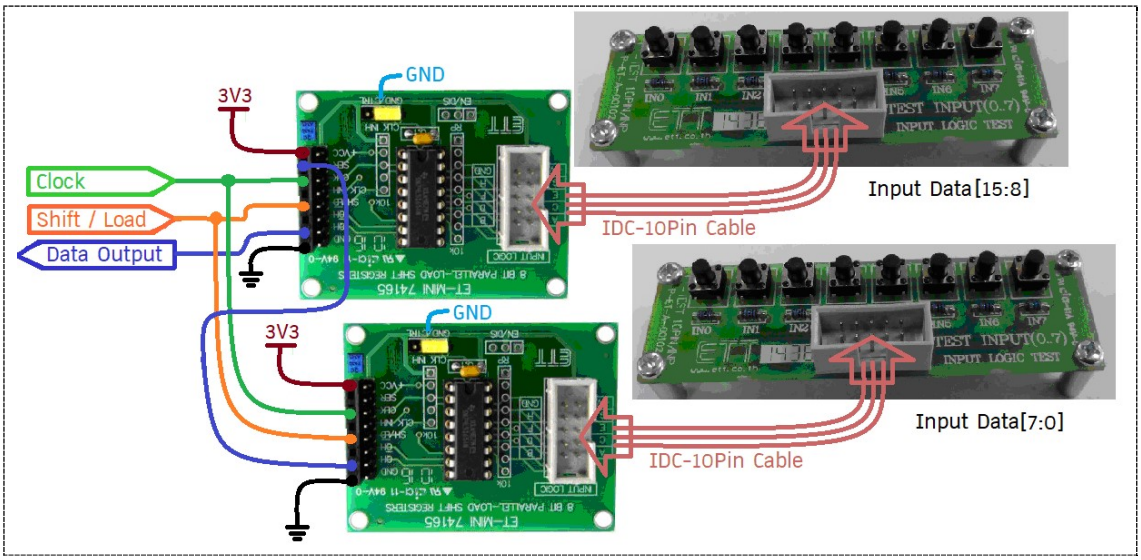
- รวมโปรแกรมข้อ 4 กับข้อ 7 เพื่อทดสอบ 8 bit Input เพื่อส่งค่าออก 8 bit Output
- พร้อมแสดงค่าที่ 4Digit 7Segment Display

8. ปรับแก้เป็น 2 IC สำหรับ 16 bit Input




```
`timescale 1ns / 1ps

module Read_16SIPO(Ser_I, Clk_I, Clk_O, SH_LD, Parl_O);
input    Clk_I, Ser_I;
output   Clk_O, SH_LD;
output   [15:0]  Parl_O;

reg [7:0] Clk_Count;     // Maximum 256 Input_Clockreg [7:0] tmp;
parameter nBit = 16;     // Using (nBit+4) Input_Clock
reg [nBit-1:0] R_Parl_O, temp_Parl_O;
reg R_ShiftLoad;

always @(posedge Clk_I) begin
         Clk_Count <= Clk_Count + 1;
         if( Clk_Count > (nBit+4)) begin                    // Over Reset Counter
                  Clk_Count       <= 0;
                  R_ShiftLoad     <= 0;
         end
         if(Clk_Count==0)        R_ShiftLoad <= 0;                 // Load
         if(Clk_Count==1)        R_ShiftLoad <= 1;                 // no Load
         if((Clk_Count>=2)&&(Clk_Count<=(nBit+1))) begin             // Shift
                  temp_Parl_O    <= temp_Parl_O << 1;
                  temp_Parl_O[0] <= Ser_I;
         end
         if(Clk_Count==(nBit+2))        R_Parl_O <= temp_Parl_O;      // Save
end

assign  Parl_O   = R_Parl_O;
assign  Clk_O    = Clk_I;
assign  SH_LD    = R_ShiftLoad;

endmodule
```

```
NET "Clk_50MHz"        LOC = P56 | IOSTANDARD = LVTTL;

NET "xCount<4>"        LOC = P134 | IOSTANDARD = LVTTL;
NET "xCount<5>"        LOC = P133 | IOSTANDARD = LVTTL;
NET "xCount<6>"        LOC = P132 | IOSTANDARD = LVTTL;
NET "xCount<7>"        LOC = P131 | IOSTANDARD = LVTTL;
NET "xCount<8>"        LOC = P127 | IOSTANDARD = LVTTL;
NET "xCount<9>"        LOC = P126 | IOSTANDARD = LVTTL;
NET "xCount<10>"       LOC = P124 | IOSTANDARD = LVTTL;
NET "xCount<11>"       LOC = P123 | IOSTANDARD = LVTTL;

NET "Data_Input"       LOC = P51 | IOSTANDARD = LVTTL ;
NET "Clk_Out"          LOC = P41 | IOSTANDARD = LVTTL ;
NET "Shift_Load"       LOC = P35 | IOSTANDARD = LVTTL ;
```



คำถาม 8 บิต

- รวมโปรแกรมข้อ 5 กับข้อ 8 เพื่อทดสอบ 16 bit Input เพื่อส่งค่าออก 16 bit Output
- พร้อมแสดงค่าที่ MAX7219 7-Segment Display

9. ทดสอบการทำงานของเซ็นเซอร์ DS18B20 Sensor

- https://github.com/douglaskastle/3S500E
- http://www.mind-tek.net/ds18b20.php
- http://narong.ece.engr.tu.ac.th/ei444/document/DS18B20_59.pdf

9.1 DS1820 Driver Verilog File

```
module ds18b20_drive(
 input       clk,
 input       rst_n,
 inout       one_wire,
 output [15:0] temperature   );


reg [5:0] cnt;

always @ (posedge clk, negedge rst_n)
  if (!rst_n)
    cnt <= 0;
  else
    if (cnt == 49)
      cnt <= 0;
    else
      cnt <= cnt + 1'b1;

reg clk_1us;

always @ (posedge clk, negedge rst_n)
  if (!rst_n)
    clk_1us <= 0;
  else
    if (cnt <= 24)
      clk_1us <= 0;
    else
      clk_1us <= 1;


reg [19:0] cnt_1us;
reg cnt_1us_clear;

always @ (posedge clk_1us)
  if (cnt_1us_clear)
    cnt_1us <= 0;
  else
    cnt_1us <= cnt_1us + 1'b1;
```

```
parameter S00     = 5'h00;
parameter S0      = 5'h01;
parameter S1      = 5'h03;
parameter S2      = 5'h02;
parameter S3      = 5'h06;
parameter S4      = 5'h07;
parameter S5      = 5'h05;
parameter S6      = 5'h04;
parameter S7      = 5'h0C;
parameter WRITE0  = 5'h0D;
parameter WRITE1  = 5'h0F;
parameter WRITE00 = 5'h0E;
parameter WRITE01 = 5'h0A;
parameter READ0   = 5'h0B;
parameter READ1   = 5'h09;
parameter READ2   = 5'h08;
parameter READ3   = 5'h18;

reg [4:0] state;


reg one_wire_buf;

reg [15:0] temperature_buf;
reg [5:0] step;
reg [3:0] bit_valid;

always @(posedge clk_1us, negedge rst_n)
begin
 if (!rst_n)
 begin
   one_wire_buf <= 1'bZ;
   step        <= 0;
   state       <= S00;
 end
 else
 begin
   case (state)
     S00 : begin
           temperature_buf <= 16'h001F;
           state          <= S0;
         end
     S0 :  begin
           cnt_1us_clear <= 1;
           one_wire_buf  <= 0;
           state        <= S1;
         end
     S1 :  begin
           cnt_1us_clear <= 0;
           if (cnt_1us == 500)
           begin
             cnt_1us_clear <= 1;
             one_wire_buf  <= 1'bZ;
             state        <= S2;
           end
         end
     S2 :  begin
           cnt_1us_clear <= 0;
           if (cnt_1us == 100)
           begin
             cnt_1us_clear <= 1;
             state        <= S3;
           end
         end
     S3 :  if (~one_wire)
           state <= S4;
         else if (one_wire)
           state <= S0;
     S4 :  begin
           cnt_1us_clear <= 0;
           if (cnt_1us == 400)
           begin
             cnt_1us_clear <= 1;
             state        <= S5;
           end
         end
     S5 :  begin
           if     (step == 0)
           begin
             step  <= step + 1'b1;
             state <= WRITE0;
           end
           else if (step == 1)
           begin
             step  <= step + 1'b1;
             state <= WRITE0;
           end
           else if (step == 2)
           begin
             one_wire_buf <= 0;
             step        <= step + 1'b1;
             state       <= WRITE01;
           end
           else if (step == 3)
           begin
             one_wire_buf <= 0;
             step        <= step + 1'b1;
```

```
        state       <= WRITE01;
      end
      else if (step == 4)
      begin
        step  <= step + 1'b1;
        state <= WRITE0;
      end
      else if (step == 5)
      begin
        step  <= step + 1'b1;
        state <= WRITE0;
      end
      else if (step == 6)
      begin
        one_wire_buf <= 0;
        step       <= step + 1'b1;
        state      <= WRITE01;
      end
      else if (step == 7)
      begin
        one_wire_buf <= 0;
        step       <= step + 1'b1;
        state      <= WRITE01;
      end

      else if (step == 8)
      begin
        step  <= step + 1'b1;
        state <= WRITE0;
      end
      else if (step == 9)
      begin
        step  <= step + 1'b1;
        state <= WRITE0;
      end
      else if (step == 10)
      begin
        one_wire_buf <= 0;
        step       <= step + 1'b1;
        state      <= WRITE01;
      end
      else if (step == 11)
      begin
        step  <= step + 1'b1;
        state <= WRITE0;
      end
      else if (step == 12)
      begin
        step  <= step + 1'b1;
        state <= WRITE0;
      end
      else if (step == 13)
      begin
        step  <= step + 1'b1;
        state <= WRITE0;
      end
      else if (step == 14)
      begin
        one_wire_buf <= 0;
        step       <= step + 1'b1;
        state      <= WRITE01;

      end
      else if (step == 15)
      begin
        step  <= step + 1'b1;
        state <= WRITE0;
      end

      else if (step == 16)
      begin
        one_wire_buf <= 1'bZ;
        step       <= step + 1'b1;
        state      <= S6;
      end


      else if (step == 17)
      begin
        step  <= step + 1'b1;
        state <= WRITE0;
      end
      else if (step == 18)
      begin
        step  <= step + 1'b1;
        state <= WRITE0;
      end
      else if (step == 19)
      begin
        one_wire_buf <= 0;
        step       <= step + 1'b1;
        state      <= WRITE01;
      end
      else if (step == 20)
      begin
        step  <= step + 1'b1;
        state <= WRITE01;
```

```
          one_wire_buf <= 0;
        end
        else if (step == 21)
        begin
          step  <= step + 1'b1;
          state <= WRITE0;
        end
        else if (step == 22)
        begin
          step  <= step + 1'b1;
          state <= WRITE0;
        end
        else if (step == 23)
        begin
          one_wire_buf <= 0;
          step       <= step + 1'b1;
          state      <= WRITE01;
        end
        else if (step == 24)
        begin
          one_wire_buf <= 0;
          step       <= step + 1'b1;
          state      <= WRITE01;
        end

        else if (step == 25)
        begin
          step  <= step + 1'b1;
          state <= WRITE0;
        end
        else if (step == 26)
        begin
          one_wire_buf <= 0;
          step       <= step + 1'b1;
          state      <= WRITE01;
        end
        else if (step == 27)
        begin
          one_wire_buf <= 0;
          step       <= step + 1'b1;
          state      <= WRITE01;
        end
        else if (step == 28)
        begin
          one_wire_buf <= 0;
          step       <= step + 1'b1;
          state      <= WRITE01;
        end
        else if (step == 29)
        begin
          one_wire_buf <= 0;
          step       <= step + 1'b1;
          state      <= WRITE01;
        end
        else if (step == 30)
        begin
          one_wire_buf <= 0;
          step       <= step + 1'b1;
          state      <= WRITE01;
        end
        else if (step == 31)
        begin
          step  <= step + 1'b1;
          state <= WRITE0;
        end
        else if (step == 32)
        begin
          one_wire_buf <= 0;
          step       <= step + 1'b1;
          state      <= WRITE01;
        end

        else if (step == 33)
        begin
          step  <= step + 1'b1;
          state <= S7;
        end
      end
    S6 :  begin
        cnt_1us_clear <= 0;
        if (cnt_1us == 750000 | one_wire)
        begin
          cnt_1us_clear <= 1;
          state       <= S0;
        end
      end

    S7 :  begin
        if     (step == 34)
        begin
          bit_valid   <= 0;
          one_wire_buf <= 0;
          step       <= step + 1'b1;
          state      <= READ0;
        end
        else if (step == 35)
        begin
```

```
          bit_valid   <= bit_valid + 1'b1;
          one_wire_buf <= 0;
          step       <= step + 1'b1;
          state      <= READ0;
        end
        else if (step == 36)
        begin
          bit_valid   <= bit_valid + 1'b1;
          one_wire_buf <= 0;
          step       <= step + 1'b1;
          state      <= READ0;
        end
        else if (step == 37)
        begin
          bit_valid   <= bit_valid + 1'b1;
          one_wire_buf <= 0;
          step       <= step + 1'b1;
          state      <= READ0;
        end
        else if (step == 38)
        begin
          bit_valid   <= bit_valid + 1'b1;
          one_wire_buf <= 0;
          step       <= step + 1'b1;
          state      <= READ0;
        end
        else if (step == 39)
        begin
          bit_valid   <= bit_valid + 1'b1;
          one_wire_buf <= 0;
          step       <= step + 1'b1;
          state      <= READ0;
        end
        else if (step == 40)
        begin
          bit_valid   <= bit_valid + 1'b1;
          one_wire_buf <= 0;
          step       <= step + 1'b1;
          state      <= READ0;
        end
        else if (step == 41)
        begin
          bit_valid   <= bit_valid + 1'b1;
          one_wire_buf <= 0;
          step       <= step + 1'b1;
          state      <= READ0;
        end
        else if (step == 42)
        begin
          bit_valid   <= bit_valid + 1'b1;
          one_wire_buf <= 0;
          step       <= step + 1'b1;
          state      <= READ0;
        end
        else if (step == 43)
        begin
          bit_valid   <= bit_valid + 1'b1;
          one_wire_buf <= 0;
          step       <= step + 1'b1;
          state      <= READ0;
        end
        else if (step == 44)
        begin
          bit_valid   <= bit_valid + 1'b1;
          one_wire_buf <= 0;
          step       <= step + 1'b1;
          state      <= READ0;
        end
        else if (step == 45)
        begin
          bit_valid   <= bit_valid + 1'b1;
          one_wire_buf <= 0;
          step       <= step + 1'b1;
          state      <= READ0;
        end
        else if (step == 46)
        begin
          bit_valid   <= bit_valid + 1'b1;
          one_wire_buf <= 0;
          step       <= step + 1'b1;
          state      <= READ0;
        end
        else if (step == 47)
        begin
          bit_valid   <= bit_valid + 1'b1;
          one_wire_buf <= 0;
          step       <= step + 1'b1;
          state      <= READ0;
        end
        else if (step == 48)
        begin
          bit_valid   <= bit_valid + 1'b1;
          one_wire_buf <= 0;
          step       <= step + 1'b1;
          state      <= READ0;
        end
        else if (step == 49)
```

```
          begin
           bit_valid    <= bit_valid + 1'b1;
           one_wire_buf <= 0;
           step        <= step + 1'b1;
           state       <= READ0;
          end
          else if (step == 50)
          begin
           step  <= 0;
           state <= S0;
          end
         end



    WRITE0 :
        begin
         cnt_1us_clear <= 0;
         one_wire_buf  <= 0;
         if (cnt_1us == 80)
         begin
          cnt_1us_clear <= 1;
          one_wire_buf  <= 1'bZ;
          state        <= WRITE00;
         end
        end
    WRITE00 :
         state <= S5;
    WRITE01 :
         state <= WRITE1;
    WRITE1 :
        begin
         cnt_1us_clear <= 0;
         one_wire_buf  <= 1'bZ;
         if (cnt_1us == 80)
         begin
          cnt_1us_clear <= 1;
          state        <= S5;
         end
        end

    READ0 : state <= READ1;
    READ1 :
        begin
         cnt_1us_clear <= 0;
         one_wire_buf  <= 1'bZ;
         if (cnt_1us == 10)
         begin
          cnt_1us_clear <= 1;
          state        <= READ2;
         end
        end
    READ2 :
        begin
         temperature_buf[bit_valid] <= one_wire;
         state                      <= READ3;
        end
    READ3 :
        begin
         cnt_1us_clear <= 0;
         if (cnt_1us == 55)
         begin
          cnt_1us_clear <= 1;
          state        <= S7;
         end
        end


    default : state <= S00;
  endcase
 end
end

assign one_wire = one_wire_buf;

wire [15:0] t_buf = temperature_buf & 16'h07FF;

assign temperature[3:0]   = (t_buf[3:0] * 10) >> 4;

assign temperature[7:4]   = (((t_buf[7:4] * 10) >> 4) >= 4'd10) ? (((t_buf[7:4] * 10) >> 4) - 'd10) : ((t_buf[7:4] * 10) >> 4);

assign temperature[11:8]  = (((t_buf[7:4] * 10) >> 4) >= 4'd10) ? (((t_buf[11:8] * 10) >> 4) + 'd1) + 'd2 : ((t_buf[11:8] * 10)
>> 4) + 'd2;

assign temperature[15:12] = temperature_buf[12] ? 1 : 0;

endmodule
```

9.2 Segment 7 Display Driver Verilog File

```
module seg_dynamic_drive(
input                   clk,
input [15:0] data,
input  [3:0] dp,

output reg [3:0] SEG_S,
output reg [7:0] SEG
);

//----------------------------------------------

reg [16:0] cnt;
always @ (posedge clk)
        cnt <= cnt + 1'b1;


reg [3:0] HEX;
always  @ (posedge clk)
begin
        case(cnt[16:15])
                2'b00 : HEX <= data[3:0];
                2'b01 : HEX <= data[7:4];
                2'b10 : HEX <= data[11:8];
                2'b11 : HEX <= data[15:12];
                default : ;
        endcase
end

//--------------drive SEG_S----------------------
//Get SEG_S through cnt[16:15]  ,every 2.6114 mS move one time
always  @ (posedge clk)
begin
        case(cnt[16:15])
                2'b00 : SEG_S <= 4'b0001;
                2'b01 : SEG_S <= 4'b0010;
                2'b10 : SEG_S <= 4'b0100;
                2'b11 : SEG_S <= 4'b1000;
                default : ;
        endcase
end

//---------drive dp------------------
always  @ (posedge clk)
begin
        case(cnt[16:15])
                2'b00 : SEG[7] <= dp[0];
                2'b01 : SEG[7] <= dp[1];
                2'b10 : SEG[7] <= dp[2];
                2'b11 : SEG[7] <= dp[3];
                default : ;
        endcase
end

//---------------------------------------
always @ (HEX)
begin
        case(HEX)
    4'h0: SEG[6:0] <= 7'b1000000;   // 0-display  1- not display
    4'h1: SEG[6:0] <= 7'b1111001;
    4'h2: SEG[6:0] <= 7'b0100100;
    4'h3: SEG[6:0] <= 7'b0110000;
    4'h4: SEG[6:0] <= 7'b0011001;
    4'h5: SEG[6:0] <= 7'b0010010;
    4'h6: SEG[6:0] <= 7'b0000010;
    4'h7: SEG[6:0] <= 7'b1111000;
    4'h8: SEG[6:0] <= 7'b0000000;
    4'h9: SEG[6:0] <= 7'b0010000;
    4'hA: SEG[6:0] <= 7'b0001000;
    4'hB: SEG[6:0] <= 7'b0000011;
    4'hC: SEG[6:0] <= 7'b1000110;
    4'hD: SEG[6:0] <= 7'b0100001;
    4'hE: SEG[6:0] <= 7'b0000110;
    4'hF: SEG[6:0] <= 7'b0001110;
    default: ;
  endcase
end


endmodule
```

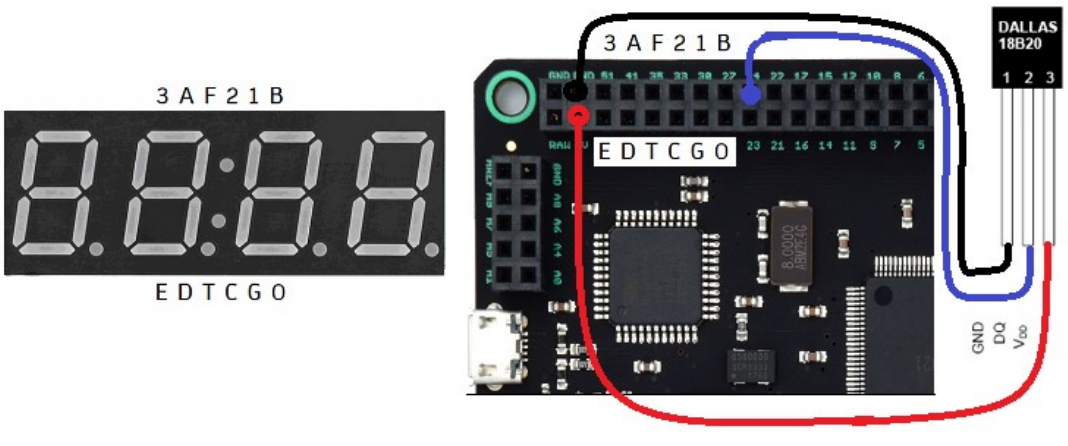
9.3 Pin I/O File

```
NET "CLOCK_50" LOC = P56 | IOSTANDARD = LVTTL ;
NET "Q_KEY"          LOC = P38 | IOSTANDARD = LVTTL ;
NET "DS18B20"  LOC = P24 ;
NET "DS18B20"        PULLUP;

NET "SEG7_SEL[3]"  LOC = P51 ;
NET "SEG7_SEL[2]"  LOC = P33 ;
NET "SEG7_SEL[1]"  LOC = P30 ;
NET "SEG7_SEL[0]"  LOC = P26 ;

NET "SEG7_SEG[7]"  LOC = P34; // dot
NET "SEG7_SEG[6]"  LOC = P29; // g
NET "SEG7_SEG[5]"  LOC = P35; // f
NET "SEG7_SEG[4]"  LOC = P50; // e
NET "SEG7_SEG[3]"  LOC = P40; // d
NET "SEG7_SEG[2]"  LOC = P32; // c
NET "SEG7_SEG[1]"  LOC = P27; // b
NET "SEG7_SEG[0]"  LOC = P41; // a
```

9.4 Top Model Verilog File

```
module ds18b20_seg7(
 input       CLOCK_50,
 input       Q_KEY,

 inout       DS18B20,

 output [7:0] SEG7_SEG,
 output [3:0] SEG7_SEL
);


wire [15:0] t_buf;

ds18b20_drive ds18b20_u0(
 .clk(CLOCK_50),
 .rst_n(Q_KEY),
 .one_wire(DS18B20),
 .temperature(t_buf)
);
seg_dynamic_drive seg7_u0(
 .clk        (CLOCK_50),

 .data       ({t_buf[11:8],t_buf[7:4],t_buf[3:0],4'h0}),
 .dp                 (4'b1011),
 .SEG        (SEG7_SEG),
 .SEG_S              (SEG7_SEL)
);
Endmodule
```

9.5 Test Circuit



9.6 คำถาม

- ปรับแก้โปรแกรมให้แสดงค่าที่ UART Serial Tx ด้วย