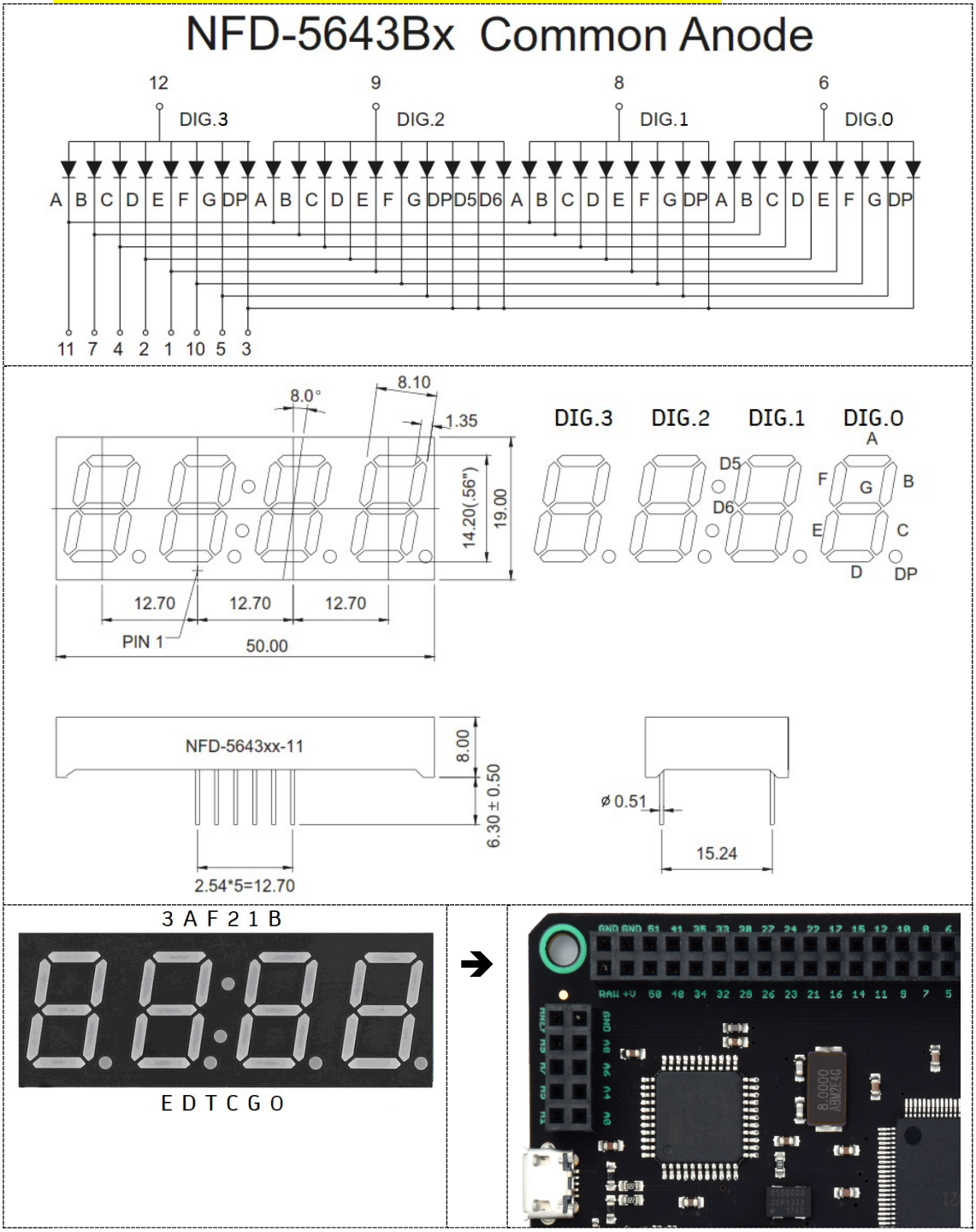
นายวิชัย ศรีสุรักษ์
B3706985

Week06_Mojo V3 – Display 7_Segment and UART Tx

1. NFD-5643Bx Common Anode 4 Digit 7 Segment Display



2. ทดสอบ Single 7 Segment Display – Main Schematic

2.1/4 Decode_7Seg Verilog File

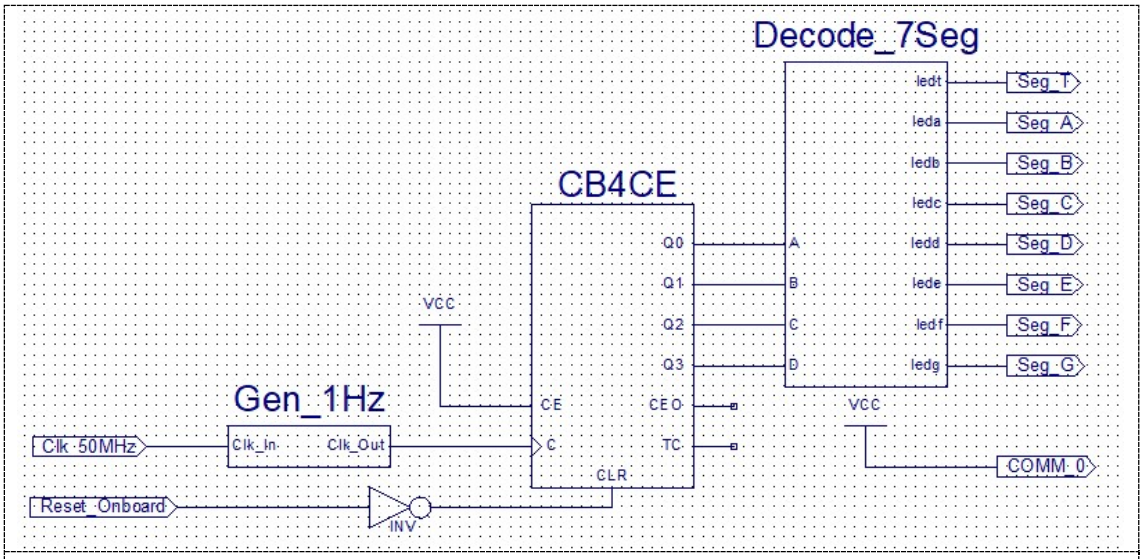
```
`timescale 1ns / 1ps

module Decode_7Seg( D, C, B, A, ledt, leda, ledb, ledc, ledd, lede, ledf,
ledg);

        input   D, C, B, A;
        output  ledt, leda, ledb, ledc, ledd, lede, ledf, ledg;
        reg     [7:0] seg_data;
        reg     [3:0] DataIn;

        always @* begin
                DataIn = { D, C, B, A };
        end

        always @(DataIn)
                case (DataIn)
                        4'b0000:        seg_data = 8'b01111110;
                        4'b0001:        seg_data = 8'b00110000;
                        4'b0010:        seg_data = 8'b01101101;
                        4'b0011:        seg_data = 8'b01111001;
                        4'b0100:        seg_data = 8'b00110011;
                        4'b0101:        seg_data = 8'b01011011;
                        4'b0110:        seg_data = 8'b01011111;
                        4'b0111:        seg_data = 8'b01110000;
                        4'b1000:        seg_data = 8'b01111111;
                        4'b1001:        seg_data = 8'b01111011;
                        4'b1010:        seg_data = 8'b01110111;
                        4'b1011:        seg_data = 8'b00011111;
                        4'b1100:        seg_data = 8'b01001110;
                        4'b1101:        seg_data = 8'b00111101;
                        4'b1110:        seg_data = 8'b01001111;
                        4'b1111:        seg_data = 8'b01000111;
                endcase

        assign ledt = ~seg_data[7]; // if Active Low(Using ~)
        assign leda = ~seg_data[6];
        assign ledb = ~seg_data[5];
        assign ledc = ~seg_data[4];
        assign ledd = ~seg_data[3];
        assign lede = ~seg_data[2];
        assign ledf = ~seg_data[1];
        assign ledg = ~seg_data[0];

endmodule
```

2.2/4 Gen_1Hz Verilog File

```
`timescale 1ns / 1ps
module Gen_1Hz( input Clk_In, output Clk_Out );
        reg rClk_Out;
        reg [27:0] Counter;
        always@(posedge Clk_In)        begin
                Counter <= Counter + 1'b1;
                if ( Counter == 25_000_000)    begin
                        Counter <= 0;
                        rClk_Out <= ~rClk_Out;
                end
        end
        assign Clk_Out = rClk_Out;
endmodule
```

50,000,000 / (2x25,000,00) = 1 Hz

2.3/4 Main Schematic File



- Generate 1 Hz — from 50MHz to 1Hz
- Decode 7 Segment — Decoder 7 Segment
- CB4CE — Counter 4 Bit
- Reset_Onboard — Reset Counter
- COMM_0 — Display on Right Digit

2.4/4 Port Input/Output ucf File

```
NET "Clk_50MHz"       LOC = P56 | IOSTANDARD = LVTTL;
NET "Reset_Onboard"   LOC = P38 | IOSTANDARD = LVTTL ;

NET "Seg_A"    LOC = P41 ;
NET "Seg_B"    LOC = P27 ;
NET "Seg_C"    LOC = P32 ;
NET "Seg_D"    LOC = P40 ;
NET "Seg_E"    LOC = P50 ;
NET "Seg_F"    LOC = P35 ;
NET "Seg_G"    LOC = P29 ;
NET "Seg_T"    LOC = P34 ;

// NET "COMM_3"       LOC = P51 ;
// NET "COMM_2"       LOC = P33 ;
// NET "COMM_1"       LOC = P30 ;
NET "COMM_0" LOC = P26 ;
```

3. ทดสอบ Single 7 Segment Display – Verilog Single File

3.1/2 Single Verilog File

```
`timescale 1ns / 1ps

module MainTest7Seg( Clk_50MHz, Reset_Onboard, Seg_A, Seg_B,
Seg_C, Seg_D, Seg_E, Seg_F, Seg_G, Seg_T, COMM_3);
        input           Reset_Onboard, Clk_50MHz;
        output  COMM_3, Seg_A, Seg_B, Seg_C, Seg_D, Seg_E, Seg_F,
Seg_G, Seg_T;

        reg     [7:0]           seg_data;
        reg     [3:0]   Counter;
        reg     [27:0]  Dly_Counter;

        always@(posedge Clk_50MHz or negedge Reset_Onboard)
        begin
                if(Reset_Onboard == 0)
                        Counter <= 0;
                else begin
                        Dly_Counter <= Dly_Counter + 1'b1;
                        if(Dly_Counter == 25_000_000)       begin
                                Dly_Counter <= 0;
                                Counter <= Counter + 1'b1;
                        end
                end
        end

        always @(Counter)
                case (Counter)
                        4'b0000:        seg_data = 8'b01111110;
                        4'b0001:        seg_data = 8'b00110000;
                        4'b0010:        seg_data = 8'b01101101;
                        4'b0011:        seg_data = 8'b01111001;
                        4'b0100:        seg_data = 8'b00110011;
                        4'b0101:        seg_data = 8'b01011011;
                        4'b0110:        seg_data = 8'b01011111;
                        4'b0111:        seg_data = 8'b01110000;
                        4'b1000:        seg_data = 8'b01111111;
                        4'b1001:        seg_data = 8'b01111011;
                        4'b1010:        seg_data = 8'b01110111;
                        4'b1011:        seg_data = 8'b00011111;
                        4'b1100:        seg_data = 8'b01001110;
                        4'b1101:        seg_data = 8'b00111101;
                        4'b1110:        seg_data = 8'b01001111;
                        4'b1111:        seg_data = 8'b01000111;
                endcase

        assign Seg_T = ~seg_data[7]; // if Active Low(Using ~)
        assign Seg_A = ~seg_data[6];
        assign Seg_B = ~seg_data[5];
        assign Seg_C = ~seg_data[4];
        assign Seg_D = ~seg_data[3];
        assign Seg_E = ~seg_data[2];
        assign Seg_F = ~seg_data[1];
        assign Seg_G = ~seg_data[0];
        assign COMM_3 = 1'b1;

endmodule
```

3.2/2 Port Input/Output ucf File

```
NET "Clk_50MHz"                 LOC = P56 | IOSTANDARD = LVTTL;
NET "Reset_Onboard"   LOC = P38 | IOSTANDARD = LVTTL ;

NET "Seg_A"    LOC = P41 ;
NET "Seg_B"    LOC = P27 ;
NET "Seg_C"    LOC = P32 ;
NET "Seg_D"    LOC = P40 ;
NET "Seg_E"    LOC = P50 ;
NET "Seg_F"    LOC = P35 ;
NET "Seg_G"    LOC = P29 ;
NET "Seg_T"    LOC = P34 ;

NET "COMM_3" LOC = P51 ;
// NET "COMM_2"       LOC = P33 ;
// NET "COMM_1"       LOC = P30 ;
// NET "COMM_0"       LOC = P26 ;
```

4. ทดสอบ 4-Digit 7 Segment Display

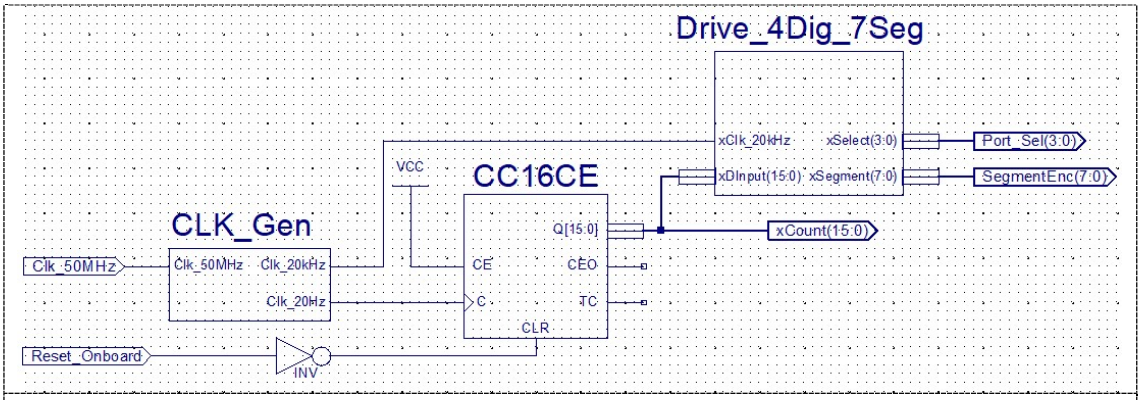
4.1/4 Clock Generate Verilog File

```
`timescale 1ns / 1ps

module CLK_Gen(input Clk_50MHz, output Clk_20Hz, Clk_20kHz );

        reg rClk_20Hz   = 1'b0;
        reg rClk_20kHz = 1'b0;
        reg [27:0] cCounter_20Hz;
        reg [27:0] cCounter_20kHz;

        always@(posedge Clk_50MHz)  begin
                cCounter_20Hz <= cCounter_20Hz + 1'b1;
                if ( cCounter_20Hz >= 1_250_000)      begin    // >>
25,000,000/20
                        cCounter_20Hz <= 0;
                        rClk_20Hz <= ~rClk_20Hz;
                end

                cCounter_20kHz <= cCounter_20kHz + 1'b1;
                if ( cCounter_20kHz >= 1_250)         begin    // >>
25,000,000/20,0000
                        cCounter_20kHz <= 0;
                        rClk_20kHz <= ~rClk_20kHz;
                end
        end

        assign Clk_20Hz         = rClk_20Hz;
        assign Clk_20kHz        = rClk_20kHz;

endmodule
```

4.2/4 Driver 4 Digit Verilog File

```
`timescale 1ns / 1ps

module Drive_4Dig_7Seg(xClk_20kHz, xDInput, xSelect, xSegment);
	input	xClk_20kHz;				// 1 Clk = 
50uSec = 0.05mS
	input		[15:0]	xDInput;
	output	[3:0]		xSelect;
	output	[7:0]	xSegment;

	parameter dStep = 100;				// 100 * 
0.05mS = 5mS
	reg	[15:0]	MasterCount;	// 4Step = 4*100 = 400 Clk
	reg	[3:0]		rSelect;
	reg	[3:0]		rEncData;
	reg	[7:0]		rSegment;

	always @(posedge xClk_20kHz) begin
		MasterCount <= MasterCount + 1'b1;
		if (MasterCount == 0*dStep)	begin
			rSelect  <= 4'b1000;
			rEncData <= xDInput[15:12];
		end
		if (MasterCount == 1*dStep)	begin
			rSelect  <= 4'b0100;
			rEncData <= xDInput[11:8];
		end
		if (MasterCount == 2*dStep)	begin
			rSelect  <= 4'b0010;
			rEncData <= xDInput[7:4];
		end
		if (MasterCount == 3*dStep)	begin
			rSelect  <= 4'b0001;
			rEncData <= xDInput[3:0];
		end
		if (MasterCount == 4*dStep)	begin
			MasterCount <= 0;
		end
	end

	always @(rEncData)
		case (rEncData)
			4'b0000:	rSegment = 8'b01111110;
			4'b0001:	rSegment = 8'b00110000;
			4'b0010:	rSegment = 8'b01101101;
			4'b0011:	rSegment = 8'b01111001;
			4'b0100:	rSegment = 8'b00110011;
			4'b0101:	rSegment = 8'b01011011;
			4'b0110:	rSegment = 8'b01011111;
			4'b0111:	rSegment = 8'b01110000;
			4'b1000:	rSegment = 8'b01111111;
			4'b1001:	rSegment = 8'b01111011;
			4'b1010:	rSegment = 8'b01110111;
			4'b1011:	rSegment = 8'b00011111;
			4'b1100:	rSegment = 8'b01001110;
			4'b1101:	rSegment = 8'b00111101;
			4'b1110:	rSegment = 8'b01001111;
			4'b1111:	rSegment = 8'b01000111;
		endcase

	assign xSelect		= rSelect;
	assign xSegment		= ~rSegment;	// not Active Low

endmodule
```

4.3/4 Main Schematic File



- CLK_Gen — Generate Clock 20Hz and 20kHz from 50MHz
- CC16CE — Counter 16 Bit
- Drive_4Dig_7Seg — Scan Display 4 Digit x 7 Segment
- Reset_Onboard — Reset Counter
- xCounter(15:0) — Counter(11:4) Monitor

4.4/4 Port Input/Output ucf File

```
NET "Clk_50MHz"              LOC = P56 | IOSTANDARD = LVTTL ;
NET "Reset_Onboard"  LOC = P38 | 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 "SegmentEnc<2>" LOC = P50 ;     // e
NET "SegmentEnc<3>" LOC = P40 ;     // d
NET "SegmentEnc<7>" LOC = P34 ;     // t
NET "SegmentEnc<4>" LOC = P32 ;     // c
NET "SegmentEnc<0>" LOC = P29 ;     // g
NET "Port_Sel<0>"     LOC = P26 ; // Coom.0

NET "Port_Sel<3>"     LOC = P51 ;     // Coom.3
NET "SegmentEnc<6>" LOC = P41 ;     // a
NET "SegmentEnc<1>" LOC = P35 ;     // f
NET "Port_Sel<2>"     LOC = P33 ;     // Coom.2
NET "Port_Sel<1>"     LOC = P30 ;     // Coom.1
NET "SegmentEnc<5>" LOC = P27 ;     // b
```

4.5 คำถาม

- หากต้องการให้แสดงการนับลงต้องทำอย่างไรบ้าง

5. ทดสอบ MAX7219 Display Control

5.1/4 test Data 32bit Verilog File

```
`timescale 1ns / 1ps

module test_Data(input Clk_50MHz, rstCount, output [31:0] oData);

	reg [31:0]	cCounter;
	reg [31:0]	roData;

	always@(posedge Clk_50MHz or negedge rstCount)	begin
		if(rstCount==0)
			roData <= 0;
		else begin
			cCounter <= cCounter + 1'b1;
			if ( cCounter == 1_250_000)	begin
				cCounter <= 0;
				roData <= roData + 1'b1;
			end
		end
	end

	assign oData = roData;

endmodule
```

25,000,000/1,250,000 = 20Hz

5.2/4 Driver MAX7219 VHDL File

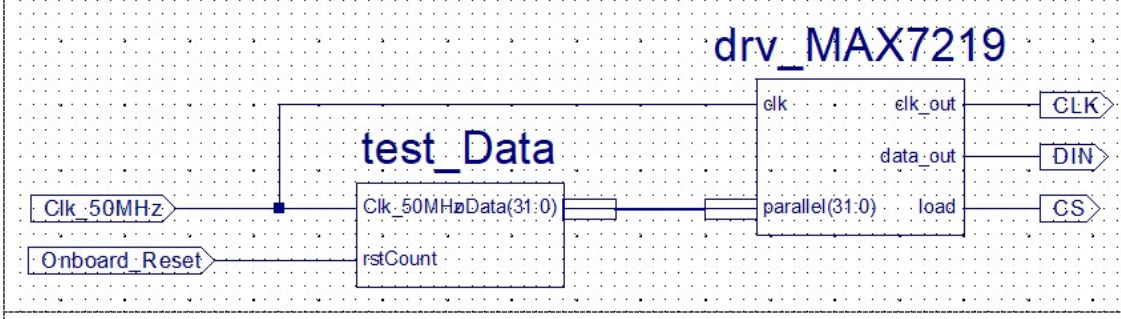
```
-- ################################################################
-- Driver for MAX7219 with 8 digit 7-segment display
-- http://stevenmerrifield.com/max7219/M7219.vhdl
-- sjm 15 May 2017
-- ################################################################
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity drv_MAX7219 is
	port (
		clk : in std_logic;
		parallel : in std_logic_vector(31 downto 0);
		clk_out : out std_logic;
		data_out : out std_logic;
		load : out std_logic
	);
end drv_MAX7219;

-- ################################################################
architecture Behavioral of drv_MAX7219 is
	attribute syn_encoding : string;

	type state_machine is (init_1, init_2, init_3, init_4, read_data, dig_7, dig_6, dig_5,
dig_4, dig_3, dig_2, dig_1, dig_0);
	attribute syn_encoding of state_machine : type is "safe";
	signal state : state_machine := init_1;

	type driver_machine is (idle, start, clk_data, clk_high, clk_low, finished);
	attribute syn_encoding of driver_machine : type is "safe";
	signal driver_state : driver_machine := idle;

	signal command : std_logic_vector(15 downto 0) := x"0000";
	signal driver_start : std_logic := '0';

-- ------------------------------------------------------------
	function hex2seg(num : std_logic_vector(3 downto 0)) return std_logic_vector is
		begin
			case num is
				when "0000" => return("01111110"); -- 0
				when "0001" => return("00110000"); -- 1
```

```
                                when "0010" => return("01101101"); -- 2
                                when "0011" => return("01111001"); -- 3
                                when "0100" => return("00110011"); -- 4
                                when "0101" => return("01011011"); -- 5
                                when "0110" => return("01011111"); -- 6
                                when "0111" => return("01110000"); -- 7
                                when "1000" => return("01111111"); -- 8
                                when "1001" => return("01111011"); -- 9
                                when "1010" => return("01110111"); -- A
                                when "1011" => return("00011111"); -- b
                                when "1100" => return("00001101"); -- c
                                when "1101" => return("00111101"); -- d
                                when "1110" => return("01001111"); -- E
                                when "1111" => return("01000111"); -- F
                                when others => return("00000000");
                                end case;
                        end hex2seg;
-- -------------------------------------------------------

-- ###########################################################
        begin -- of Behavioral
                process
                        variable counter : integer := 0;
                        variable clk_counter : integer := 0;
                        variable latch_in : std_logic_vector(31 downto 0) := x"00000000";
                        variable dig0_data : std_logic_vector(7 downto 0) := x"00";
                        variable dig1_data : std_logic_vector(7 downto 0) := x"00";
                        variable dig2_data : std_logic_vector(7 downto 0) := x"00";
                        variable dig3_data : std_logic_vector(7 downto 0) := x"00";
                        variable dig4_data : std_logic_vector(7 downto 0) := x"00";
                        variable dig5_data : std_logic_vector(7 downto 0) := x"00";
                        variable dig6_data : std_logic_vector(7 downto 0) := x"00";
                        variable dig7_data : std_logic_vector(7 downto 0) := x"00";

-- =============================================================
                        begin -- of process
-- -------------------------------------------------------
                                wait until rising_edge(clk);
-- -------------------------------------------------------
                                case state is
                                        when init_1 =>
            if (driver_state = idle) then
              command <= x"0c01"; -- shutdown / normal operation
              driver_state <= start;
              state <= init_2;
            end if;
          when init_2 =>
            if (driver_state = idle) then
              command <= x"0900"; -- decode mode
              driver_state <= start;
              state <= init_3;
            end if;
          when init_3 =>
            if (driver_state = idle) then
              command <= x"0A04"; -- intensity
              driver_state <= start;
              state <= init_4;
            end if;
          when init_4 =>
            if (driver_state = idle) then
              command <= x"0B07"; -- scan limit
              driver_state <= start;
              state <= read_data;
            end if;
          when read_data =>
              latch_in := parallel;
              dig7_data := hex2seg(latch_in(31 downto 28));
              dig6_data := hex2seg(latch_in(27 downto 24));
              dig5_data := hex2seg(latch_in(23 downto 20));
              dig4_data := hex2seg(latch_in(19 downto 16));
              dig3_data := hex2seg(latch_in(15 downto 12));
              dig2_data := hex2seg(latch_in(11 downto 8));
              dig1_data := hex2seg(latch_in(7 downto 4));
```

```
            dig0_data := hex2seg(latch_in(3 downto 0));
            state <= dig_7;
        when dig_7 =>
          if (driver_state = idle) then
            command <= x"08" & dig7_data;
            driver_state <= start;
            state <= dig_6;
          end if;
        when dig_6 =>
          if (driver_state = idle) then
            command <= x"07" & dig6_data;
            driver_state <= start;
            state <= dig_5;
          end if;
        when dig_5 =>
          if (driver_state = idle) then
            command <= x"06" & dig5_data;
            driver_state <= start;
            state <= dig_4;
          end if;
        when dig_4 =>
          if (driver_state = idle) then
            command <= x"05" & dig4_data;
            driver_state <= start;
            state <= dig_3;
          end if;
        when dig_3 =>
          if (driver_state = idle) then
            command <= x"04" & dig3_data;
            driver_state <= start;
            state <= dig_2;
          end if;
        when dig_2 =>
          if (driver_state = idle) then
            command <= x"03" & dig2_data;
            driver_state <= start;
            state <= dig_1;
          end if;
        when dig_1 =>
          if (driver_state = idle) then
            command <= x"02" & dig1_data;
            driver_state <= start;
            state <= dig_0;
          end if;
        when dig_0 =>
          if (driver_state = idle) then
            command <= x"01" & dig0_data;
            driver_state <= start;
            state <= read_data;
          end if;
        when others => null;
                                end case;
-- ------------------------------------------------------------
                                if (clk_counter < 100) then
                                        clk_counter := clk_counter + 1;
                                else
                                        clk_counter := 0;
                                        case driver_state is
                                                when idle =>
                                                        load <= '1';
                                                        clk_out <= '0';
                                                when start =>
                                                        load <= '0';
                                                        counter := 16;
                                                        driver_state <= clk_data;
                                                when clk_data =>
                                                        counter := counter - 1;
                                                        data_out <= command(counter);
                                                        driver_state <= clk_high;
                                                when clk_high =>
                                                        clk_out <= '1';
                                                        driver_state <= clk_low;
                                                when clk_low =>
```

```
                                                clk_out <= '0';
                                                if (counter = 0) then
                                                        load <= '1';
                                                        driver_state <= finished;
                                                else
                                                        driver_state <= clk_data;
                                                end if;
                                        when finished =>
                                                driver_state <= idle;
                                        when others => null;
                                end case;
                        end if; -- clk_counter
-- ------------------------------------------------------
        end process;
-- ============================================================
end Behavioral;
-- ############################################################
```

5.3/4 Main Schematic File



- Test_Data — Generate data 32bit at 20Hz
- Drv_MAX7219 — Driver MAX7219 Display Control
- Reset_Onboard — Reset Counter

5.4/4 Port Input/Output ucf File

```
NET "Clk_50MHz"                 LOC = P56 | IOSTANDARD = LVTTL ;
NET "Onboard_Reset"   LOC = P38 | IOSTANDARD = LVTTL ;

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

NET "DIN"       LOC = P51 ;
NET "CLK"       LOC = P41 ;
NET "CS"        LOC = P35 ;
```

5.5 คำถาม

- หากต้องการให้แสดงการนับแบบ BCD ต้องปรับแก้อะไรบ้าง

6. Serial Communication – UART Tx

- https://embeddedmicro.com/blogs/tutorials/hello-world
- https://embeddedmicro.com/blogs/tutorials/asynchronous-serial
- https://codereview.stackexchange.com/questions/115003/verilog-uart-transmitter

Asynchronous serial communication, often shortened to just serial, is one of the easiest ways to communicate between two different devices. In it's simplest form, it consists of just two connections. One line for sending data and the other for receiving data.

There are many variations on the classic serial bus, but this tutorial will cover just the basics. You should be able to communicate with most serial devices including a computer.

Baud Rate

As the name implies, this protocol is asynchronous. All that means is that there is no shared clock. To get around not having a clock, both devices need to agree to the rate that data can be transmitted. The rate that data is sent is known as the baud rate. The unit for baud rate is bits/sec and this indirectly sets the width of each bit.

In theory, you can use any baud rate that you like. However, to make it easier to setup devices, there are a handful of standard baud rates. In most cases you will be using one of the following rates.

Baud Rates (bits/sec) ➔ 4800, 9600, 14400, 19200, 28800, 38400, 56000, 57600, 115200, 128000, 153600, 230400, 256000

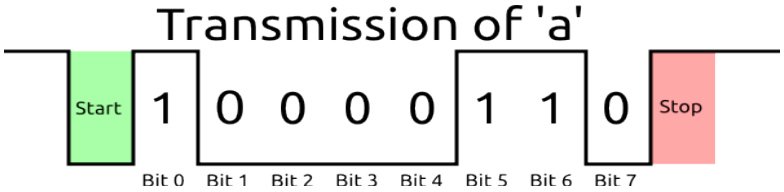
Start and Stop Bits

Another side effect of not having a clock is the need for start and stops bits. A serial port typically sends data out in packets of 8 bits, or a byte. Since it is asynchronous, you never know when the other device will send a byte!

To let the receiver know you are going to send a byte, a start bit is sent. This is simply a single bit with the value of 0.

Similarly, after the byte has been sent out, a stop bit is sent. Typically one stop bit is sent, but sometimes two are sent. Stop bits have the value of 1.

To better understand what goes on, take a look at this example transmission.



This is a transmission of the character 'a' which has an ASCII value of 97, or a binary value of 01100001.

The serial line idles high, meaning when nothing is being sent it is held at the value 1. This is why the start bit is 0, so the line transitions from high to low. This gives an indication that the line is active.

You may have noticed that order of the bits is reversed. That is because the LSB (least-significant bit) is typically transmitted first.

6.1/4 UART Sending Verilog File

```
//////////////////////////////////////////////////////////////////
module uart_send #     (       parameter BAUD_RATE = 115200,
                               parameter CLOCK_SPEED_MHZ = 50)
                       (       input [7:0] data_byte,
                               input start_send,
                               input clk,
                               output tx,
                               output ready);

        parameter integer CYCLES_WAIT = CLOCK_SPEED_MHZ * 1e6 / BAUD_RATE;

        parameter IDLE = 0;
        parameter START_BIT = 1;
        parameter END_BIT = 2;
        parameter DATA_BIT = 3;

        reg [2:0] state = IDLE;
        reg [15:0] cycle_count = 0;
        reg [3:0] bit_index = 0;
        reg [7:0] data;

        assign tx = state == IDLE ? 1 :
                        state == START_BIT ? 0 :
                        state == END_BIT ? 1 :
                        data[bit_index];

        assign ready = state == IDLE;

        always @(posedge clk) begin
                if(state != IDLE)
                        data <= data_byte;
                if(cycle_count == CYCLES_WAIT) cycle_count <= 0;
                else cycle_count <= cycle_count + 1;
                if(state == IDLE && start_send) begin
                        state <= START_BIT;
                        cycle_count <= 0;
                end else if(state == START_BIT && cycle_count == CYCLES_WAIT) begin
                        state <= DATA_BIT;
                        bit_index <= 0;
                end else if(state == DATA_BIT && cycle_count == CYCLES_WAIT) begin
                        if(bit_index == 7) state <= END_BIT;
                        else bit_index <= bit_index + 1;
                end else if(state == END_BIT && cycle_count == CYCLES_WAIT) begin
                        state <= IDLE;
                end
        end

endmodule
```

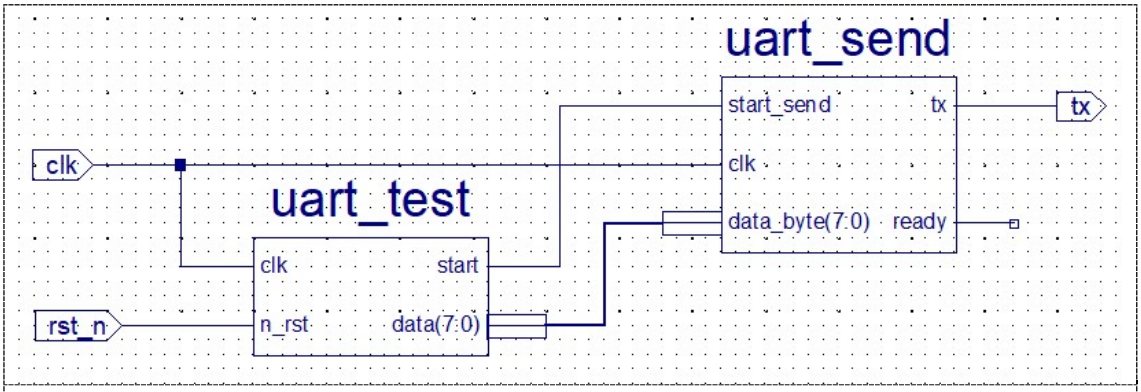
2.4/4 Port Input/Output ucf File

```
#Created by Constraints Editor (xc6slx9-tqg144-3) - 2012/11/05
NET "clk" TNM_NET = clk;
TIMESPEC TS_clk = PERIOD "clk" 50 MHz HIGH 50%;

NET "clk"       LOC = P56       | IOSTANDARD = LVTTL;
NET "rst_n"     LOC = P38       | IOSTANDARD = LVTTL;

NET "tx"        LOC = P51       | IOSTANDARD = LVTTL;
```

6.2/4 UART Test Verilog File

```
//////////////////////////////////////////////////////////////////////
module uart_test     #(parameter DelayStep=10000000)
                  ( input clk, n_rst, output start, output [7:0] data );
      parameter ASCII_Start = "A";
      parameter ASCII_Stop  = "Z";
      reg [31:0]       count = 0;
      reg [7:0]  rData = ASCII_Start;
      reg [7:0]  cData = ASCII_Start;
      reg              rStart = 0;

      always @(posedge clk or negedge n_rst)
            if (n_rst==0)
                  begin
                        cData <= ASCII_Start;
                        rData <= ASCII_Start;
                  end
            else
                  begin
                        if (count == DelayStep)
                              begin
                                    count <= 0;
                                    cData <= cData + 1'b1;
                                    if(cData <= ASCII_Stop)              rData <= cData;
                                    else if(cData == (ASCII_Stop+1))     rData <= 8'b00001101;
                                    else if(cData == (ASCII_Stop+2))     rData <= 8'b00001010;
                                    else cData <= ASCII_Start;
                              end
                        else
                              begin
                                    count <= count + 1'b1;
                                    if (count == 100)      rStart <= 1;     else      rStart <= 0;
                              end
                  end

      assign data = rData;
      assign start = rStart;

endmodule
```

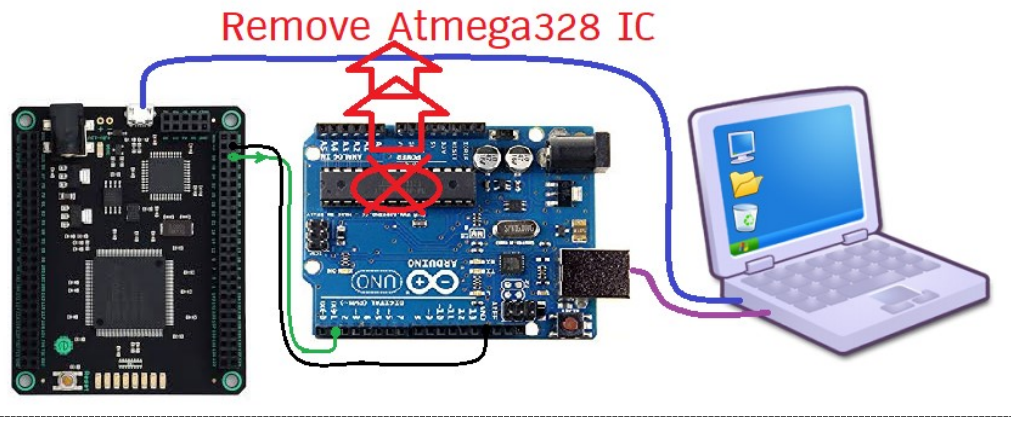
6.3/4 Main Schematic File



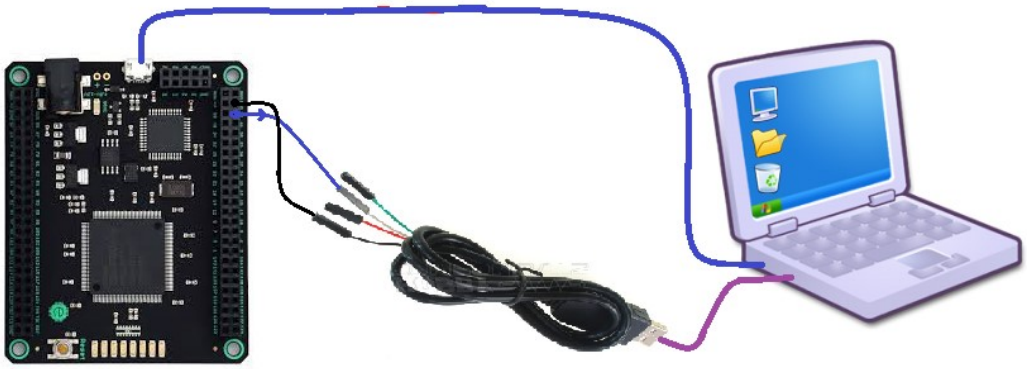
- Uart_send — UART Send module
- Uart_test — Data for test UART
- Reset_Onboard — Reset Counter

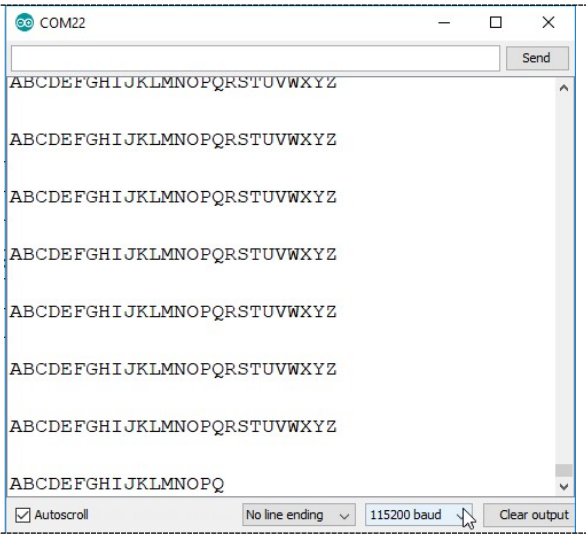
6.5 วงจรทดสอบ

แบบที่ 1: ใช้ Arduino Board → จำเป็นต้องถอดไอซี Atmega328 ออกก่อนทดสอบ



แบบที่ 2: ใช้ USB to RS232-TTL Board < [5V,Tx,Rx,Gnd] = [Red,Grn,Wht,Blk] >





ทดสอบด้วย Serial Monitor ที่อยู่กับ Arduino IDE ด้วย Baud rate = 115200

6.5 คำถาม

- หากต้องการให้แสดงชื่อและรหัสนักศึกษาของตัวเอง ต้องปรับแก้อะไรบ้าง

7. UART Echo (Tx, Rx)

7.1 Pin I/O File

```
NET "clk"          LOC = P56 | IOSTANDARD = LVTTL ;
NET "rst_n"        LOC = P38 | IOSTANDARD = LVTTL ;

NET "rs232_tx"     LOC = P51 | IOSTANDARD = LVTTL ;
NET "rs232_rx"     LOC = P41 | IOSTANDARD = LVTTL ;
```

7.2 UART_1of3 Schematic File

```
module speed_select(
                                        clk,rst_n,
                                        bps_start,clk_bps
                                    );

input clk;  // 50MHz???
input rst_n;          //???????
input bps_start;      //?????????????????
output clk_bps;       // clk_bps??????????????????????

/*
parameter             bps9600   = 5207,   //????9600bps
                                        bps19200 = 2603,   //????19200bps
                                        bps38400 = 1301,   //????38400bps
                                        bps57600 = 867,    //????57600bps
                                        bps115200          = 433;     //????115200bps

parameter             bps9600_2         = 2603,
                                        bps19200_2         = 1301,
                                        bps38400_2         = 650,
                                        bps57600_2         = 433,
                                        bps115200_2 = 216;
*/

          //??????????????????????????
`define               BPS_PARA          5207      //????9600???????
`define     BPS_PARA_2                  2603      //????9600?????????????????

reg[12:0] cnt;                          //????
reg clk_bps_r;                          //????????

//-----------------------------------------------------------
reg[2:0] uart_ctrl;    // uart????????
//-----------------------------------------------------------

always @ (posedge clk or negedge rst_n)
          if(!rst_n) cnt <= 13'd0;
          else if((cnt == `BPS_PARA) || !bps_start) cnt <= 13'd0;        //???????
          else cnt <= cnt+1'b1;                                          //?????????

always @ (posedge clk or negedge rst_n)
          if(!rst_n) clk_bps_r <= 1'b0;
//???????&& bps_start?
          else if(cnt == `BPS_PARA_2 && bps_start) clk_bps_r <= 1'b1; // clk_bps_r???????????????,???????????????
          else clk_bps_r <= 1'b0;

assign clk_bps = clk_bps_r;

endmodule
```

7.3 UART_2of3 Schematic File

```
module my_uart_rx( clk,rst_n,  rs232_rx, rx_data, rx_int, clk_bps,bps_start );

input clk;              // 50MHz???
input rst_n;            //???????
input rs232_rx;         // RS232??????
input clk_bps;          // clk_bps??????????????????????
output bps_start;                 //????????????????????
output[7:0] rx_data; //??????????????????
output rx_int;          //????????,?????????????

//-------------------------------------------------------------------
reg rs232_rx0,rs232_rx1,rs232_rx2,rs232_rx3;       //???????????
wire neg_rs232_rx;  //???????????

always @ (posedge clk or negedge rst_n) begin
            if(!rst_n) begin
                              rs232_rx0 <= 1'b0;
                              rs232_rx1 <= 1'b0;
                              rs232_rx2 <= 1'b0;
                              rs232_rx3 <= 1'b0;
                        end
            else begin
                              rs232_rx0 <= rs232_rx;
                              rs232_rx1 <= rs232_rx0;
                              rs232_rx2 <= rs232_rx1;
                              rs232_rx3 <= rs232_rx2;
                        end
end
            //?????????????<20ns-40ns???(???????????)?
            //?????????????????????????????????????????
            //??????????????????????40ns??
assign neg_rs232_rx = rs232_rx3 & rs232_rx2 & ~rs232_rx1 & ~rs232_rx0;          //???????neg_rs232_rx????????

//-------------------------------------------------------------------
reg bps_start_r;
reg[3:0] num;           //????
reg rx_int;             //????????,?????????????

always @ (posedge clk or negedge rst_n)
            if(!rst_n) begin
                              bps_start_r <= 1'bz;
                              rx_int <= 1'b0;
                        end
            else if(neg_rs232_rx) begin                 //????????rs232_rx????????
                              bps_start_r <= 1'b1;//??????????
                              rx_int <= 1'b1;                               //??????????
                        end
            else if(num==4'd11) begin                   //?????????          ///??????num????????11.???12??
                              bps_start_r <= 1'b0;//???????????????
                              rx_int <= 1'b0;                               //??????????
                        end

assign bps_start = bps_start_r;

//-------------------------------------------------------------------
reg[7:0] rx_data_r;               //?????????????????????
//-------------------------------------------------------------------

reg[7:0] rx_temp_data;            //?????????

always @ (posedge clk or negedge rst_n)
            if(!rst_n) begin
                              rx_temp_data <= 8'd0;
                              num <= 4'd0;
                              rx_data_r <= 8'd0;
                        end
            else if(rx_int) begin  //??????
                        if(clk_bps) begin     //???????,???????????8bit???1?2????
                                          num <= num+1'b1;
                                          case (num)
                                                            4'd1: rx_temp_data[0] <= rs232_rx;       //???0bit
                                                            4'd2: rx_temp_data[1] <= rs232_rx;       //???1bit
                                                            4'd3: rx_temp_data[2] <= rs232_rx;       //???2bit
                                                            4'd4: rx_temp_data[3] <= rs232_rx;       //???3bit
                                                            4'd5: rx_temp_data[4] <= rs232_rx;       //???4bit
                                                            4'd6: rx_temp_data[5] <= rs232_rx;       //???5bit
                                                            4'd7: rx_temp_data[6] <= rs232_rx;       //???6bit
                                                            4'd8: rx_temp_data[7] <= rs232_rx;       //????7bit
                                                            default: ;
                                                  endcase
                              end
                        else if(num == 4'd11) begin
            //?????????????1+8+1(2)=11bit?????///??????num????????11.???12??
                                          num <= 4'd0;                                  //???STOP????,num??
                                          rx_data_r <= rx_temp_data;    //???????????rx_data?
                              end
                        end

assign rx_data = rx_data_r;

endmodule
```

7.4 UART_3of3 Verilog File

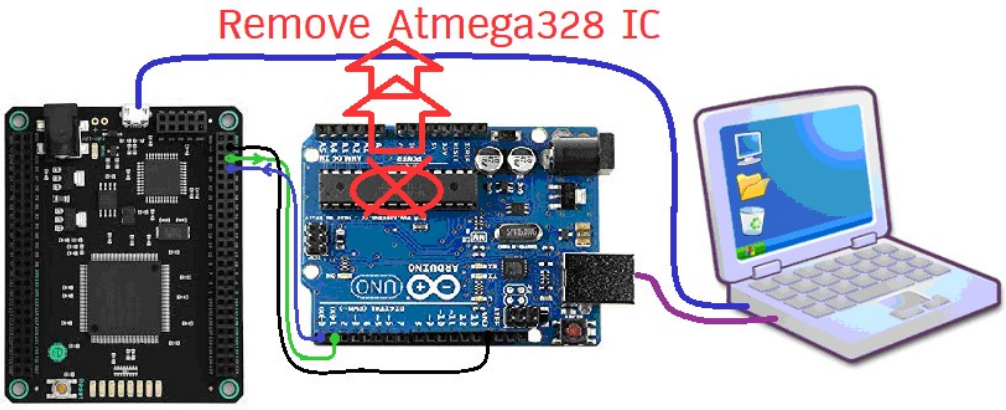
```
module my_uart_tx(
                                        clk,rst_n,
                                        rx_data,rx_int,rs232_tx,
                                        clk_bps,bps_start
                                );

input clk;                      // 50MHz???
input rst_n;                    //???????
input clk_bps;                  // clk_bps_r???????????????,???????????????
input[7:0] rx_data;   //???????
input rx_int;                   //????????,?????????????,??????????????????????
output rs232_tx;      // RS232??????
output bps_start;     //???????????????????????

//------------------------------------------------------------
reg rx_int0,rx_int1,rx_int2;      //rx_int?????????????
wire neg_rx_int;      // rx_int??????

always @ (posedge clk or negedge rst_n) begin
            if(!rst_n) begin
                                rx_int0 <= 1'b0;
                                rx_int1 <= 1'b0;
                                rx_int2 <= 1'b0;
                        end
            else begin
                                rx_int0 <= rx_int;
                                rx_int1 <= rx_int0;
                                rx_int2 <= rx_int1;
                        end
end

assign neg_rx_int =  ~rx_int1 & rx_int2;   //????????neg_rx_int???????????

//------------------------------------------------------------
reg[7:0] tx_data;     //?????????
//------------------------------------------------------------
reg bps_start_r;
reg tx_en; //????????????
reg[3:0] num;

always @ (posedge clk or negedge rst_n) begin
            if(!rst_n) begin
                                bps_start_r <= 1'bz;
                                tx_en <= 1'b0;
                                tx_data <= 8'd0;
                        end
            else if(neg_rx_int) begin         //???????????????????
                                bps_start_r <= 1'b1;
                                tx_data <= rx_data; //???????????????????
                                tx_en <= 1'b1;                    //?????????
                        end
            else if(num==4'd11) begin      //?????????
                                bps_start_r <= 1'b0;
                                tx_en <= 1'b0;
                        end
end

assign bps_start = bps_start_r;

//------------------------------------------------------------
reg rs232_tx_r;

always @ (posedge clk or negedge rst_n) begin
            if(!rst_n) begin
                                num <= 4'd0;
                                rs232_tx_r <= 1'b1;
                        end
            else if(tx_en) begin
                                if(clk_bps)             begin
                                                        num <= num+1'b1;
                                                        case (num)
                                                                4'd0: rs232_tx_r <= 1'b0;       //?????
                                                                4'd1: rs232_tx_r <= tx_data[0];           //??bit0
                                                                4'd2: rs232_tx_r <= tx_data[1];           //??bit1
                                                                4'd3: rs232_tx_r <= tx_data[2];           //??bit2
                                                                4'd4: rs232_tx_r <= tx_data[3];           //??bit3
                                                                4'd5: rs232_tx_r <= tx_data[4];           //??bit4
                                                                4'd6: rs232_tx_r <= tx_data[5];           //??bit5
                                                                4'd7: rs232_tx_r <= tx_data[6];           //??bit6
                                                                4'd8: rs232_tx_r <= tx_data[7];           //??bit7
                                                                4'd9: rs232_tx_r <= 1'b1;       //?????
                                                                default: rs232_tx_r <= 1'b1;
                                                                endcase
                                                        end
                                else if(num==4'd11) num <= 4'd0;         //??
                        end
end

assign rs232_tx = rs232_tx_r;

endmodule
```

7.5 Top Module Verilog File

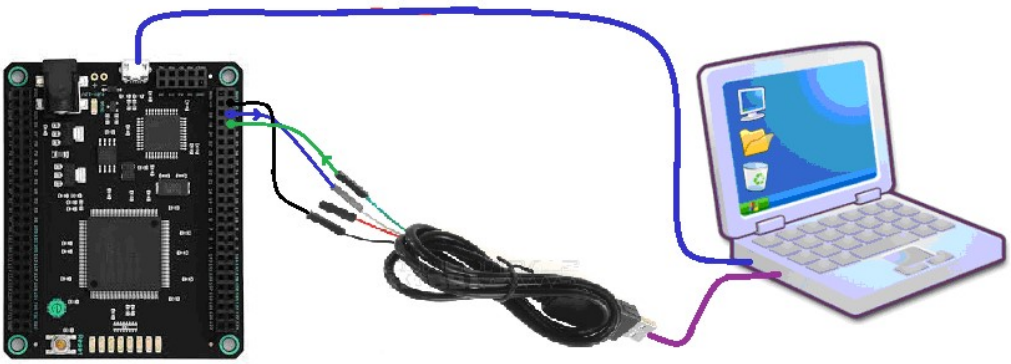
```
module uart( clk, rst_n, rs232_rx, rs232_tx );

input clk;                      // 50MHz???
input rst_n;                    //???????

input rs232_rx;                 // RS232??????
output rs232_tx;     //         RS232??????

wire bps_start1,bps_start2;     //??????????????????
wire clk_bps1,clk_bps2;                 // clk_bps_r???????????????,???????????????
wire[7:0] rx_data;   //??????????????????
wire rx_int;                    //????????,?????????????
//---------------------------------------------------
//?????????speed_rx?speed_tx??????????????????????
//????????????????????????????
////////////////////////////////////////////
speed_select                    speed_rx(
                                                        .clk(clk),  //???????
                                                        .rst_n(rst_n),
                                                        .bps_start(bps_start1),
                                                        .clk_bps(clk_bps1)
                                                );

my_uart_rx                              my_uart_rx(
                                                        .clk(clk),  //??????
                                                        .rst_n(rst_n),
                                                        .rs232_rx(rs232_rx),
                                                        .rx_data(rx_data),
                                                        .rx_int(rx_int),
                                                        .clk_bps(clk_bps1),
                                                        .bps_start(bps_start1)
                                                );

////////////////////////////////////////////
speed_select                    speed_tx(
                                                        .clk(clk),  //???????
                                                        .rst_n(rst_n),
                                                        .bps_start(bps_start2),
                                                        .clk_bps(clk_bps2)
                                                );

my_uart_tx                              my_uart_tx(
                                                        .clk(clk),  //??????
                                                        .rst_n(rst_n),
                                                        .rx_data(rx_data),
                                                        .rx_int(rx_int),
                                                        .rs232_tx(rs232_tx),
                                                        .clk_bps(clk_bps2),
                                                        .bps_start(bps_start2)
                                                );

Endmodule
```

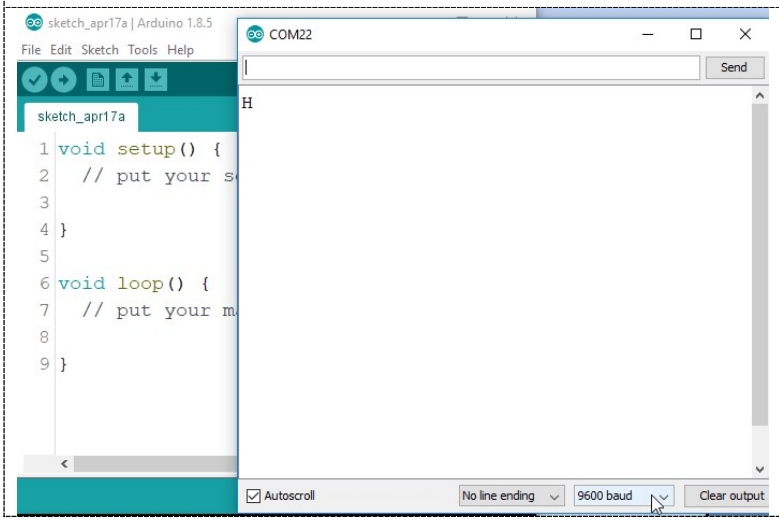
7.6 วงจรทดสอบ

แบบที่ 1: ใช้ Arduino Board → จำเป็นต้องถอดไอซี Atmega328 ออกก่อนทดสอบ



แบบที่ 2: ใช้ USB to RS232-TTL Board < [5V,Tx,Rx,Gnd] = [Red,Grn,Wht,Blk] >





Baud rate = 9600

7.7 คำถาม

- หากต้องการให้แสดงค่า ASCII Code ที่รับได้บน LED 8 ตัวด้วยต้องปรับแก้โปรแกรมอย่างไร