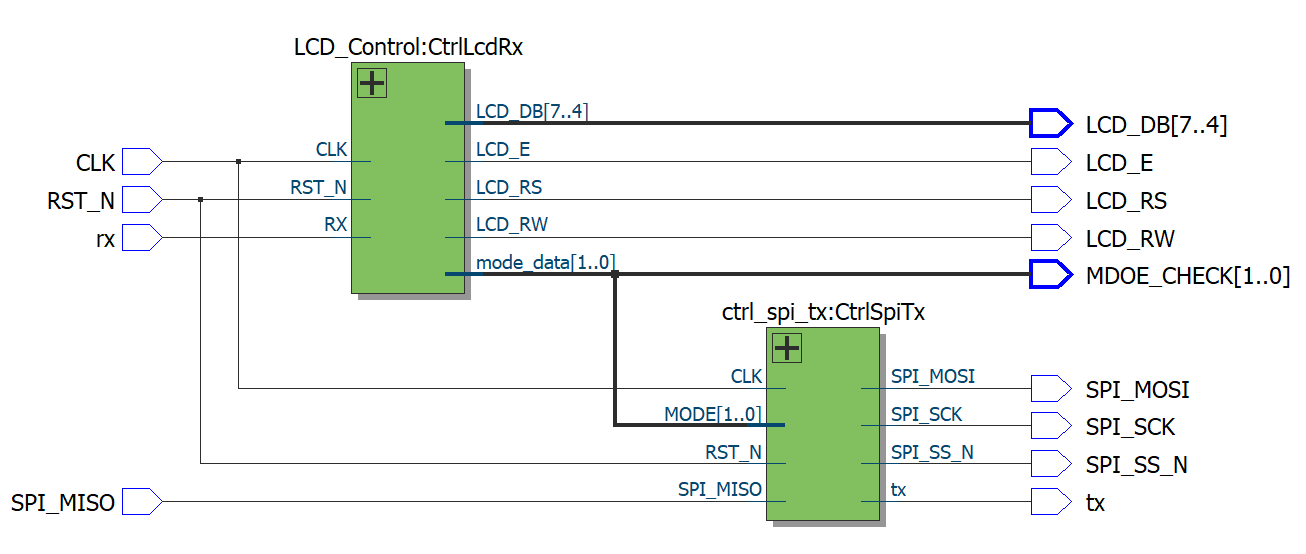
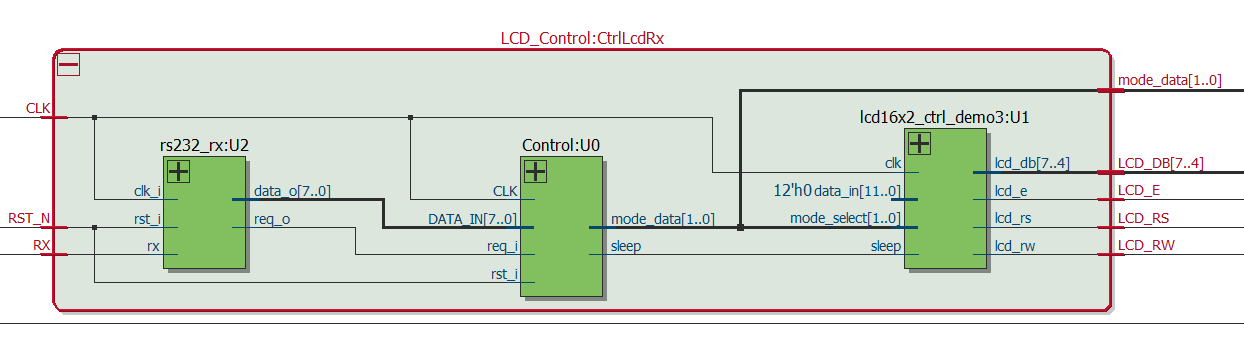
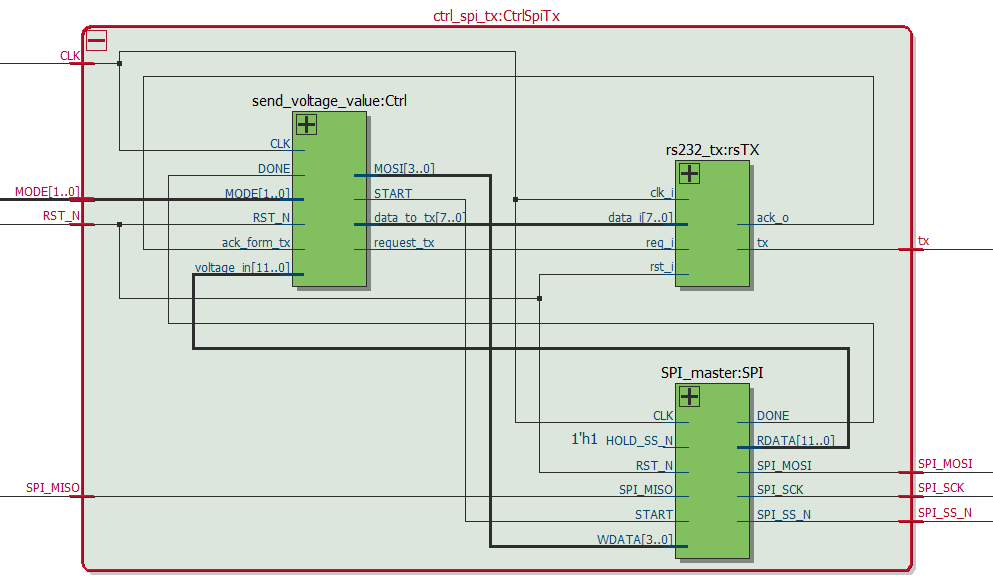
**ภาคผนวก**

รูปภาพ ภ.1 ภาพ Block Diagram ของโปรแกรมโดยรวม

รูปภาพ ภ.2 ภาพ Block Diagram ของส่วนควบคุมข้อมูลและ LCD



รูปภาพ ภ.3 ภาพ Block Diagram ของส่วนอ่านค่า, แปลงข้อมูลและส่งข้อมูล

**โค้ด VHDL ส่วนหลัก**

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.numeric\_std.all;

entity VHDL\_Oscilloscope is

port(

CLK, RST\_N, rx :in std\_logic;

tx : out std\_logic;

MDOE\_CHECK : out std\_logic\_vector(1 downto 0);

-- LCD Module

LCD\_E : out std\_logic;

LCD\_RS : out std\_logic;

LCD\_RW : out std\_logic;

LCD\_DB : out std\_logic\_vector(7 downto 4);

--SPI module

SPI\_SCK : out std\_logic;

SPI\_MOSI : out std\_logic;

SPI\_MISO : in std\_logic;

SPI\_SS\_N : out std\_logic );

end VHDL\_Oscilloscope;

architecture behave of VHDL\_Oscilloscope is

component LCD\_Control is

port( CLK : in std\_logic;

RST\_N : in std\_logic; -- Reset

RX : in std\_logic;

-- LCD Module

LCD\_E : out std\_logic;

LCD\_RS : out std\_logic;

LCD\_RW : out std\_logic;

LCD\_DB : out std\_logic\_vector(7 downto 4);

mode\_data : out std\_logic\_vector(1 downto 0));

end component;

component ctrl\_spi\_tx is

port(

CLK, RST\_N : in std\_logic;

MODE : in std\_logic\_vector(1 downto 0);

tx : out std\_logic;

--SPI module

SPI\_SCK : out std\_logic;

SPI\_MOSI : out std\_logic;

SPI\_MISO : in std\_logic;

SPI\_SS\_N : out std\_logic );

end component;

signal mode\_data : std\_logic\_vector(1 downto 0);

begin

CtrlLcdRx : LCD\_Control

port map(CLK, RST\_N, rx, LCD\_E, LCD\_RS, LCD\_RW, LCD\_DB, mode\_data);

CtrlSpiTx : ctrl\_spi\_tx

port map(CLK, RST\_N, mode\_data, tx, SPI\_SCK, SPI\_MOSI, SPI\_MISO,SPI\_SS\_N);

MDOE\_CHECK <= mode\_data;

end behave;

**โค้ดส่วนควบคุมการทำงานและ LCD**

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.numeric\_std.all;

entity LCD\_Control is

port( CLK : in std\_logic;

RST\_N : in std\_logic; -- Reset

RX : in std\_logic;

-- LCD Module

LCD\_E : out std\_logic;

LCD\_RS : out std\_logic;

LCD\_RW : out std\_logic;

LCD\_DB : out std\_logic\_vector(7 downto 4);

mode\_data : out std\_logic\_vector(1 downto 0));

end LCD\_Control;

architecture behavior of LCD\_Control is

component Control is

port ( CLK : in std\_logic;

req\_i : in std\_logic;

rst\_i : in std\_logic; -- synchronous reset, active low

mode\_data : out std\_logic\_vector(1 downto 0);

sleep : out std\_logic:='0';

DATA\_IN : in std\_logic\_vector(7 downto 0));

end component;

component lcd16x2\_ctrl\_demo3 is

port (

clk, sleep : in std\_logic;

data\_in : in std\_logic\_vector(11 downto 0);

mode\_select : in std\_logic\_vector(1 downto 0);

lcd\_e : out std\_logic;

lcd\_rs : out std\_logic;

lcd\_rw : out std\_logic;

lcd\_db : out std\_logic\_vector(7 downto 4));

end component;

component rs232\_rx is

generic(

SYSTEM\_SPEED : integer := 50e6;

BAUDRATE : integer := 460800);

port( clk\_i : in std\_logic; -- system clock

rst\_i : in std\_logic; -- synchronous reset, active high

req\_o : out std\_logic := '0'; -- Rx req

data\_o : out std\_logic\_vector(7 downto 0); -- Rx data

rx : in std\_logic); -- Rx input

end component;

signal mode\_data\_lcd : std\_logic\_vector(1 downto 0);

signal channel\_data : std\_logic\_vector(3 downto 0);

signal signal\_data : std\_logic\_vector(11 downto 0);

signal data\_temp : std\_logic\_vector(7 downto 0);

signal rx\_out : std\_logic;

signal start\_spi : std\_logic;

signal done\_spi : std\_logic;

signal HOLD\_SS\_N : std\_logic; -- status CS logic before active

signal sleep : std\_logic;

begin

U0:Control

port map(CLK, rx\_out, RST\_N, mode\_data\_lcd, sleep, data\_temp);

U1:lcd16x2\_ctrl\_demo3

port map(CLK, sleep, signal\_data, mode\_data\_lcd, LCD\_E, LCD\_RS, LCD\_RW, LCD\_DB);

U2:rs232\_rx

port map(CLK, RST\_N, rx\_out, data\_temp, RX);

mode\_data <= mode\_data\_lcd;

end behavior;

**โค้ดส่วนรับข้อมูลจาก Pc**

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.numeric\_std.all;

use ieee.math\_real.all;

entity rs232\_rx is

generic(

system\_speed : integer := 50e6; -- clock speed, in Hz

baudrate : integer := 9600); -- baudrate

port(

clk\_i : in std\_logic; -- system clock

req\_o : out std\_logic;

rst\_i : in std\_logic; -- synchronous reset, active high

data\_o : out std\_logic\_vector(7 downto 0); -- Rx data

rx : in std\_logic); -- Rx input

end rs232\_rx;

architecture behave of rs232\_rx is

constant max\_count: integer := (system\_speed/baudrate);

type state\_type is (

wait\_for\_rx\_start,

wait\_half\_bit,

receive\_bits,

wait\_for\_stop\_bit);

signal state : state\_type := wait\_for\_rx\_start;

signal baudrate\_counter : integer range 0 to max\_count := 0;

signal bit\_counter : integer range 0 to 7 := 0;

signal shift\_register : std\_logic\_vector(7 downto 0) := (others => '0');

begin

process(clk\_i)

begin

if rising\_edge(clk\_i) then

if rst\_i = '0' then

state <= wait\_for\_rx\_start;

data\_o <= (others => '0');

req\_o <= '1';

else

case state is

when wait\_for\_rx\_start =>

req\_o <= '1';

if rx = '0' then

-- start bit received, wait for a half bit time

-- to sample bits in the middle of the signal

state <= wait\_half\_bit;

baudrate\_counter <= (max\_count/2)-1;

end if;

when wait\_half\_bit =>

if baudrate\_counter = 0 then

-- now we are in the middle of the start bit,

-- wait a full bit for the middle of the first bit

state <= receive\_bits;

bit\_counter <= 7;

baudrate\_counter <= max\_count-1;

else

baudrate\_counter <= baudrate\_counter-1;

end if;

when receive\_bits =>

-- sample a bit

if baudrate\_counter = 0 then

shift\_register <= rx & shift\_register(7 downto 1);

if bit\_counter = 0 then

state <= wait\_for\_stop\_bit;

else

bit\_counter <= bit\_counter-1;

end if;

baudrate\_counter <= max\_count-1;

else

baudrate\_counter <= baudrate\_counter-1;

end if;

when wait\_for\_stop\_bit =>

-- wait for the middle of the stop bit

if baudrate\_counter = 0 then

state <= wait\_for\_rx\_start;

if rx = '1' then

data\_o <= shift\_register;

req\_o <= '0';

-- else: missing stop bit, ignore

end if;

else

baudrate\_counter <= baudrate\_counter-1;

end if;

end case;

end if;

end if;

end process;

end behave;

**โค้ดส่วนตรวจสอบข้อมูลที่ได้รับมา**

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_arith.all;

use ieee.std\_logic\_unsigned.all;

entity Control is

port (

CLK : in std\_logic;

req\_i : in std\_logic;

rst\_i : in std\_logic; -- synchronous reset, active low

mode\_data : out std\_logic\_vector(1 downto 0);

sleep : out std\_logic:='0';

DATA\_IN : in std\_logic\_vector(7 downto 0));

end Control;

architecture behave of Control is

constant TIME\_SLEEP : integer := 150000000;

type state\_type is (

check\_start\_bit,

check\_mode\_bit

);

signal state : state\_type := check\_start\_bit;

signal count : integer := 0;

begin

process(CLK, req\_i, rst\_i)

begin

if rst\_i = '0' then

state <= check\_start\_bit;

elsif rising\_edge(CLK)then

if count > TIME\_SLEEP then

mode\_data <= "00";

sleep <= '1';

count <= 0;

end if;

case state is

when check\_start\_bit =>

count <= count + 1;

if req\_i = '0' and Data\_IN = x"23" then -- # ascii

sleep <= '0';

count <= 0;

state <= check\_mode\_bit;

end if;

when check\_mode\_bit =>

if req\_i = '0' then

if DATA\_IN = x"30" then -- 0 ascii

mode\_data <= "00"; -- Off Mode

state <= check\_start\_bit;

elsif DATA\_IN = x"31" then -- 1 ascii

mode\_data <= "01"; -- SingleCh1 Mode

state <= check\_start\_bit;

elsif DATA\_IN = x"32" then -- 2 ascii

mode\_data <= "10"; -- SingleCh2 Mode

state <= check\_start\_bit;

elsif DATA\_IN = x"33" then -- 3 ascii

mode\_data <= "11"; -- DualChannel Mode

state <= check\_start\_bit;

end if;

end if;

end case;

end if;

end process;

end behave;

**โค้ดส่วนควบคุมการแสดงผลบน LCD**

library ieee;

use ieee.numeric\_std.all;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_unsigned.all;

-------------------------------------------------------------------------------

entity lcd16x2\_ctrl\_demo3 is

port (

clk, sleep : in std\_logic;

data\_in : in std\_logic\_vector(11 downto 0);

mode\_select : in std\_logic\_vector(1 downto 0);

lcd\_e : out std\_logic;

lcd\_rs : out std\_logic;

lcd\_rw : out std\_logic;

lcd\_db : out std\_logic\_vector(7 downto 4));

end entity lcd16x2\_ctrl\_demo3;

-------------------------------------------------------------------------------

architecture behavior of lcd16x2\_ctrl\_demo3 is

--

signal timer : natural range 0 to 100000000 := 0;

signal switch\_lines : std\_logic := '0';

signal line1 : std\_logic\_vector(127 downto 0);

signal line2 : std\_logic\_vector(127 downto 0);

-- component generics

constant CLK\_PERIOD\_NS : positive := 10; -- 100 Mhz

-- component ports

signal rst : std\_logic;

signal line1\_buffer : std\_logic\_vector(127 downto 0);

signal line2\_buffer : std\_logic\_vector(127 downto 0);

-- value of input

signal count : integer range 0 to 10 := 0;

begin -- architecture behavior

-- component instantiation

DUT : entity work.lcd16x2\_ctrl

generic map (

CLK\_PERIOD\_NS => CLK\_PERIOD\_NS)

port map (

clk => clk,

rst => rst,

lcd\_e => lcd\_e,

lcd\_rs => lcd\_rs,

lcd\_rw => lcd\_rw,

lcd\_db => lcd\_db,

line1\_buffer => line1\_buffer,

line2\_buffer => line2\_buffer);

line1(127 downto 120) <= X"4f"; -- O

line1(119 downto 112) <= X"73"; -- s

line1(111 downto 104) <= X"63"; -- c

line1(103 downto 96) <= X"69"; -- i

line1(95 downto 88) <= X"6c"; -- l

line1(87 downto 80) <= X"6c"; -- l

line1(79 downto 72) <= X"6f"; -- o

line1(71 downto 64) <= X"73"; -- s

line1(63 downto 56) <= X"63"; -- c

line1(55 downto 48) <= X"6f"; -- o

line1(47 downto 40) <= X"70"; -- p

line1(39 downto 32) <= X"65"; -- e

line1(31 downto 24) <= X"20"; -- " "

line1(23 downto 16) <= X"32"; -- 2

line1(15 downto 8) <= X"43"; -- C

line1(7 downto 0) <= X"48"; -- H

rst <= '0';

-- switch lines every second

process(clk)

begin

if rising\_edge(clk) then

if timer = 0 then

timer <= 25000000;

if sleep = '1' then

line2(127 downto 120) <= X"20"; -- " "

line2(119 downto 112) <= X"20"; -- " "

line2(111 downto 104) <= X"20"; -- " "

line2(103 downto 96) <= X"4d"; -- M

line2(95 downto 88) <= X"6f"; -- o

line2(87 downto 80) <= X"64"; -- d

line2(79 downto 72) <= X"65"; -- e

line2(71 downto 64) <= X"3a"; -- :

line2(63 downto 56) <= X"53"; -- S

line2(55 downto 48) <= X"6c"; -- l

line2(47 downto 40) <= X"65"; -- e

line2(39 downto 32) <= X"65"; -- e

line2(31 downto 24) <= X"70"; -- p

line2(23 downto 16) <= X"20"; -- " "

line2(15 downto 8) <= X"20"; -- " "

line2(7 downto 0) <= X"20"; -- " "

elsif sleep = '0' then

if mode\_select = "00" then

line2(127 downto 120) <= X"20"; -- " "

line2(119 downto 112) <= X"20"; -- " "

line2(111 downto 104) <= X"20"; -- " "

line2(103 downto 96) <= X"20"; -- " "

line2(95 downto 88) <= X"4d"; -- M

line2(87 downto 80) <= X"6f"; -- o

line2(79 downto 72) <= X"64"; -- d

line2(71 downto 64) <= X"65"; -- e

line2(63 downto 56) <= X"3a"; -- :

line2(55 downto 48) <= X"4f"; -- O

line2(47 downto 40) <= X"66"; -- f

line2(39 downto 32) <= X"66"; -- f

line2(31 downto 24) <= X"20"; -- " "

line2(23 downto 16) <= X"20"; -- " "

line2(15 downto 8) <= X"20"; -- " "

line2(7 downto 0) <= X"20"; -- " "

elsif mode\_select = "01" then

line2(127 downto 120) <= X"20"; -- " "

line2(119 downto 112) <= X"4d"; -- M

line2(111 downto 104) <= X"6f"; -- o

line2(103 downto 96) <= X"64"; -- d

line2(95 downto 88) <= X"65"; -- e

line2(87 downto 80) <= X"3a"; -- :

line2(79 downto 72) <= X"53"; -- S

line2(71 downto 64) <= X"69"; -- i

line2(63 downto 56) <= X"6e"; -- n

line2(55 downto 48) <= X"67"; -- g

line2(47 downto 40) <= X"6c"; -- g

line2(39 downto 32) <= X"65"; -- e

line2(31 downto 24) <= X"43"; -- C

line2(23 downto 16) <= X"48"; -- H

line2(15 downto 8) <= X"31"; -- 1

line2(7 downto 0) <= X"20"; -- " "

elsif mode\_select = "10" then

line2(127 downto 120) <= X"20"; -- " "

line2(119 downto 112) <= X"4d"; -- M

line2(111 downto 104) <= X"6f"; -- o

line2(103 downto 96) <= X"64"; -- d

line2(95 downto 88) <= X"65"; -- e

line2(87 downto 80) <= X"3a"; -- :

line2(79 downto 72) <= X"53"; -- S

line2(71 downto 64) <= X"69"; -- i

line2(63 downto 56) <= X"6e"; -- n

line2(55 downto 48) <= X"67"; -- g

line2(47 downto 40) <= X"6c"; -- g

line2(39 downto 32) <= X"65"; -- e

line2(31 downto 24) <= X"43"; -- C

line2(23 downto 16) <= X"48"; -- H

line2(15 downto 8) <= X"32"; -- 2

line2(7 downto 0) <= X"20"; -- " "

elsif mode\_select = "11" then

line2(127 downto 120) <= X"4d"; -- M

line2(119 downto 112) <= X"6f"; -- o

line2(111 downto 104) <= X"64"; -- d

line2(103 downto 96) <= X"65"; -- e

line2(95 downto 88) <= X"3a"; -- :

line2(87 downto 80) <= X"44"; -- D

line2(79 downto 72) <= X"75"; -- u

line2(71 downto 64) <= X"61"; -- a

line2(63 downto 56) <= X"6c"; -- l

line2(55 downto 48) <= X"43"; -- C

line2(47 downto 40) <= X"68"; -- h

line2(39 downto 32) <= X"61"; -- a

line2(31 downto 24) <= X"6e"; -- n

line2(23 downto 16) <= X"6e"; -- n

line2(15 downto 8) <= X"65"; -- e

line2(7 downto 0) <= X"6c"; -- l

end if;

end if;

else

timer <= timer - 1;

end if;

end if;

end process;

line1\_buffer <= line1; --when switch\_lines = '1' else line1;

line2\_buffer <= line2; --when switch\_lines = '1' else line2;

end architecture behavior;

**โค้ดส่วนการแปลงค่าและส่งข้อมูล**

library ieee;

use ieee.std\_logic\_1164.all;

entity ctrl\_spi\_tx is

port(

CLK, RST\_N : in std\_logic;

MODE : in std\_logic\_vector(1 downto 0);

tx : out std\_logic;

SPI\_SCK : out std\_logic;

SPI\_MOSI : out std\_logic;

SPI\_MISO : in std\_logic;

--SHOW : out std\_logic\_vector(7 downto 0);

SPI\_SS\_N : out std\_logic );

end ctrl\_spi\_tx;

architecture behave of ctrl\_spi\_tx is

component SPI\_master is

generic( WAIT\_COUNT\_MAX : integer := 25 ;--1/2 SPI clk period

data\_M\_to\_S: integer := 4; --data send master to slave\*\*\*\*\*\*\*\*\*\*\*\*\*\*

data\_S\_to\_M: integer := 12);--data send slave to master\*\*\*\*\*\*\*\*\*\*\*\*\*\*

port(

CLK : in std\_logic; --clock input

RST\_N : in std\_logic; --Reset

START : in std\_logic; --Start

HOLD\_SS\_N : in std\_logic; --status CS logic before active

WDATA : in std\_logic\_vector( data\_M\_to\_S-1 downto 0 ); --data writed master to slave

RDATA : out std\_logic\_vector( data\_S\_to\_M-1 downto 0 ) := (others => '0'); --data resived slave to master

DONE : out std\_logic; --

-- SPI interface

SPI\_SCK : out std\_logic;

SPI\_MOSI : out std\_logic;

SPI\_MISO : in std\_logic;

SPI\_SS\_N : out std\_logic );

end component;

component rs232\_tx is

generic(

SYSTEM\_SPEED : integer := 50e6; -- clock speed, in Hz

BAUDRATE : integer := 460800); -- baudrate

port(

clk\_i : in std\_logic; -- system clock

rst\_i : in std\_logic; -- synchronous reset, active-Low

req\_i : in std\_logic; -- Tx request

ack\_o : out std\_logic; -- Tx acknowledge

data\_i : in std\_logic\_vector(7 downto 0); -- Tx data

tx : out std\_logic ); -- Tx output

end component;

component send\_voltage\_value is

port(

CLK, RST\_N : in std\_logic;

--RX

MODE : in std\_logic\_vector(1 downto 0);

-- SPI

voltage\_in : in std\_logic\_vector(11 downto 0);

MOSI : out std\_logic\_vector(3 downto 0);

START : out std\_logic:='0';

DONE : in std\_logic;

--TX

data\_to\_tx : out std\_logic\_vector(7 downto 0);

request\_tx : out std\_logic:='0';

ack\_form\_tx : in std\_logic);

end component;

signal HOLD\_SS\_N : std\_logic:='1'; -- status CS logic before active

signal WDATA : std\_logic\_vector( 3 downto 0 ); -- data writed master to slave

signal RDATA : std\_logic\_vector( 11 downto 0 ) := (others => '0'); -- data resived slave to master

signal DONE, START, request, ack, to\_tx : std\_logic;

signal data\_to\_tx : std\_logic\_vector(7 downto 0);

begin

--SHOW(1 downto 0) <= MODE;

--SHOW(7 downto 4) <= WDATA;

--SHOW2 <= ack;

SPI:SPI\_master

port map(CLK, RST\_N, START, HOLD\_SS\_N, WDATA, RDATA, DONE, SPI\_SCK, SPI\_MOSI, SPI\_MISO, SPI\_SS\_N);

rsTX:rs232\_tx

port map(CLK, RST\_N, request, ack, data\_to\_tx, to\_tx);

Ctrl:send\_voltage\_value

port map(CLK, RST\_N, MODE, RDATA, WDATA, START, DONE, data\_to\_tx, request, ack);

tx<=to\_tx;

--SHOW2 <= to\_tx;

end behave;

**โค้ดส่วนการอ่านค่าและแปลงค่าเพื่อเตรียมการส่ง**

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.numeric\_std.all;

use ieee.std\_logic\_unsigned.all;

entity send\_voltage\_value is

port(

CLK, RST\_N : in std\_logic;

--RX

MODE : in std\_logic\_vector(1 downto 0);

-- SPI

voltage\_in : in std\_logic\_vector(11 downto 0);

MOSI : out std\_logic\_vector(3 downto 0);

START : out std\_logic:='0';

DONE : in std\_logic;

--TX

data\_to\_tx : out std\_logic\_vector(7 downto 0);

--check : out std\_logic\_vector(7 downto 0);

request\_tx : out std\_logic:='0';

ack\_form\_tx : in std\_logic);

end send\_voltage\_value;

architecture behave of send\_voltage\_value is

constant SEND\_PER\_SEC : integer := 6000;

signal MAX\_COUNTER : integer := 50e6/SEND\_PER\_SEC;

constant VOLTAGE\_REFER : integer := 3300;

type state\_type is (

WAIT\_TIME\_REQ, -- Default state

--MODE SINGLE

REQ\_VOLTAGE\_SINGLE,

WAIT\_DONE\_SINGLE,

SEND\_MODE\_SINGLE,

WAIT\_SEND\_MODE\_SINGLE,

SEND\_VOLTAGE\_PART1\_SINGLE,

WAIT\_SEND\_VOLTAGE\_PART1\_SINGLE,

SEND\_VOLTAGE\_PART2\_SINGLE,

WAIT\_SEND\_VOLTAGE\_PART2\_SINGLE,

--MODE DUAL

REQ\_VOLTAGE\_CH1\_DUAL,

WAIT\_DONE\_CH1\_DUAL,

REQ\_VOLTAGE\_CH2\_DUAL,

WAIT\_DONE\_CH2\_DUAL,

SEND\_MODE\_DUAL,

WAIT\_SEND\_MODE\_DUAL,

SEND\_VOLTAGE\_CH1\_PART1\_DUAL,

WAIT\_SEND\_VOLTAGE\_CH1\_PART1\_DUAL,

SEND\_VOLTAGE\_CH1\_PART2\_DUAL,

WAIT\_SEND\_VOLTAGE\_CH1\_PART2\_DUAL,

SEND\_VOLTAGE\_CH2\_PART1\_DUAL,

WAIT\_SEND\_VOLTAGE\_CH2\_PART1\_DUAL,

SEND\_VOLTAGE\_CH2\_PART2\_DUAL,

WAIT\_SEND\_VOLTAGE\_CH2\_PART2\_DUAL);

signal state: state\_type := WAIT\_TIME\_REQ;

signal count\_time, voltage\_data, voltage\_data2 : integer :=0;

signal mode\_buffer : std\_logic\_vector(1 downto 0) := "00";

signal voltage\_buffer : std\_logic\_vector(11 downto 0);

begin

process(CLK) begin

if rising\_edge(CLK) then

count\_time <= count\_time + 1;

case state is

when WAIT\_TIME\_REQ =>

request\_tx <= '1';

START <= '0';

mode\_buffer <= MODE; -- update MODE\_buffer

if count\_time > MAX\_COUNTER then

count\_time <= 0;

----------------

-- IF Single mode

if mode\_buffer = "01" or mode\_buffer = "10" then

state <= REQ\_VOLTAGE\_SINGLE;

-- IF Dual mode

elsif mode\_buffer = "11" then

state <= REQ\_VOLTAGE\_CH1\_DUAL;

end if;

end if;

----------------------------------------------------------------------------------------

----------------------------------------------------------------------------------------

--MODE SINGLE

when REQ\_VOLTAGE\_SINGLE =>

START <= '1';

if MODE = "01" then

MOSI <= "1101";

elsif MODE = "10" then

MOSI <= "1111";

end if;

state <= WAIT\_DONE\_SINGLE;

when WAIT\_DONE\_SINGLE =>

START <= '0';

if DONE = '1' then

voltage\_data <= to\_integer(unsigned(voltage\_in)) \* VOLTAGE\_REFER / 4095;

state <= SEND\_MODE\_SINGLE;

end if;

----------------------------------------------------------------------------------------

when SEND\_MODE\_SINGLE =>

data\_to\_tx <= "100000" & mode\_buffer;

request\_tx <= '0';

state <= WAIT\_SEND\_MODE\_SINGLE;

when WAIT\_SEND\_MODE\_SINGLE =>

request\_tx <= '1';

if ack\_form\_tx = '1' then

state <= SEND\_VOLTAGE\_PART1\_SINGLE;

end if;

---------------------------------

when SEND\_VOLTAGE\_PART1\_SINGLE =>

data\_to\_tx <= std\_logic\_vector(to\_unsigned(voltage\_data/100, 8));

request\_tx <= '0';

state <= WAIT\_SEND\_VOLTAGE\_PART1\_SINGLE;

when WAIT\_SEND\_VOLTAGE\_PART1\_SINGLE =>

request\_tx <= '1';

if ack\_form\_tx = '1' then

state <= SEND\_VOLTAGE\_PART2\_SINGLE;

end if;

--

when SEND\_VOLTAGE\_PART2\_SINGLE =>

data\_to\_tx <= std\_logic\_vector(to\_unsigned(voltage\_data mod 100, 8));

request\_tx <= '0';

state <= WAIT\_SEND\_VOLTAGE\_PART2\_SINGLE;

when WAIT\_SEND\_VOLTAGE\_PART2\_SINGLE =>

request\_tx <= '1';

if ack\_form\_tx = '1' then

state <= WAIT\_TIME\_REQ;

end if;

----------------------------------------------------------------------------------------

----------------------------------------------------------------------------------------

--MODE DUAL

when REQ\_VOLTAGE\_CH1\_DUAL => -- Request Voltage from SPI

MOSI <= "1101";

START <= '1';

if DONE = '0' then

state <= WAIT\_DONE\_CH1\_DUAL;

end if;

when WAIT\_DONE\_CH1\_DUAL => --

START <= '0';

if DONE = '1' then

voltage\_data <= to\_integer(unsigned(voltage\_in)) \* VOLTAGE\_REFER / 4095;

state <= REQ\_VOLTAGE\_CH2\_DUAL;

end if;

when REQ\_VOLTAGE\_CH2\_DUAL =>

MOSI <= "1111";

START <= '1';

if DONE = '0' then

state <= WAIT\_DONE\_CH2\_DUAL;

end if;

when WAIT\_DONE\_CH2\_DUAL =>

START <= '0';

if DONE = '1' then

voltage\_data2 <= to\_integer(unsigned(voltage\_in)) \* VOLTAGE\_REFER / 4095;

state <= SEND\_MODE\_DUAL;

end if;

----------------------------------------------------------------------------------------

when SEND\_MODE\_DUAL =>

data\_to\_tx <= "100000" & mode\_buffer;

request\_tx <= '0';

state <= WAIT\_SEND\_MODE\_DUAL;

when WAIT\_SEND\_MODE\_DUAL =>

request\_tx <= '1';

if ack\_form\_tx = '1' then

state <= SEND\_VOLTAGE\_CH1\_PART1\_DUAL;

end if;

----------------------------------------------------------------------------------------

when SEND\_VOLTAGE\_CH1\_PART1\_DUAL =>

data\_to\_tx <= std\_logic\_vector(to\_unsigned(voltage\_data/100, 8));

request\_tx <= '0';

state <= WAIT\_SEND\_VOLTAGE\_CH1\_PART1\_DUAL;

when WAIT\_SEND\_VOLTAGE\_CH1\_PART1\_DUAL =>

request\_tx <= '1';

if ack\_form\_tx = '1' then

state <= SEND\_VOLTAGE\_CH1\_PART2\_DUAL;

end if;

when SEND\_VOLTAGE\_CH1\_PART2\_DUAL =>

data\_to\_tx <= std\_logic\_vector(to\_unsigned(voltage\_data mod 100, 8));

request\_tx <= '0';

state <= WAIT\_SEND\_VOLTAGE\_CH1\_PART2\_DUAL;

when WAIT\_SEND\_VOLTAGE\_CH1\_PART2\_DUAL =>

request\_tx <= '1';

if ack\_form\_tx = '1' then

state <= SEND\_VOLTAGE\_CH2\_PART1\_DUAL;

end if;

----------------------------------------------------------------------------------------

when SEND\_VOLTAGE\_CH2\_PART1\_DUAL =>

data\_to\_tx <= std\_logic\_vector(to\_unsigned(voltage\_data2/100, 8));

request\_tx <= '0';

state <= WAIT\_SEND\_VOLTAGE\_CH2\_PART1\_DUAL;

when WAIT\_SEND\_VOLTAGE\_CH2\_PART1\_DUAL =>

request\_tx <= '1';

if ack\_form\_tx = '1' then

state <= SEND\_VOLTAGE\_CH2\_PART2\_DUAL;

end if;

when SEND\_VOLTAGE\_CH2\_PART2\_DUAL =>

data\_to\_tx <= std\_logic\_vector(to\_unsigned(voltage\_data2 mod 100, 8));

request\_tx <= '0';

state <= WAIT\_SEND\_VOLTAGE\_CH2\_PART2\_DUAL;

when WAIT\_SEND\_VOLTAGE\_CH2\_PART2\_DUAL =>

request\_tx <= '1';

if ack\_form\_tx = '1' then

state <= WAIT\_TIME\_REQ;

end if;

end case;

end if;

end process;

end behave;

**โค้ดส่วนอ่านค่าแรงดัน Analog จาก MCP3202 ( SPI )**

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.numeric\_std.all;

entity SPI\_master is

generic( WAIT\_COUNT\_MAX : integer := 25 ;--1/2 SPI clk period

data\_M\_to\_S: integer := 4; --data send master to slave\*\*\*\*\*\*\*\*\*\*\*\*\*\*

data\_S\_to\_M: integer := 12);--data send slave to master\*\*\*\*\*\*\*\*\*\*\*\*\*\*

port(

CLK : in std\_logic; --clock input

RST\_N : in std\_logic; --Reset

START : in std\_logic; --Start

HOLD\_SS\_N : in std\_logic; --status CS logic before active

WDATA : in std\_logic\_vector( data\_M\_to\_S-1 downto 0 ); --data writed master to slave

RDATA : out std\_logic\_vector( data\_S\_to\_M-1 downto 0 ) := (others => '0'); --data resived slave to master

DONE : out std\_logic; --

-- SPI interface

SPI\_SCK : out std\_logic;

SPI\_MOSI : out std\_logic;

SPI\_MISO : in std\_logic;

SPI\_SS\_N : out std\_logic );

end SPI\_master;

architecture behave of SPI\_master is

type state\_type is (S0,S1,S2,S3,S4,S5,S6,S7,S8);

signal state : state\_type := S0;

-- signal bit\_count : integer range 0 to 7 := 0;

signal wait\_count : integer range 0 to 100 := 0;

signal sreg : std\_logic\_vector( data\_M\_to\_S-1 downto 0 ) ;

signal rreg : std\_logic\_vector( data\_S\_to\_M downto 0 ) ;--\*\*null bit

signal spi\_cs\_n : std\_logic := '1';

signal index\_send : integer range 0 to 15 ;

signal index\_resive : integer range 0 to 15 ;

begin

process ( RST\_N, CLK )

begin

if RST\_N = '0' then -- active-low reset

SPI\_MOSI <= '1';

SPI\_SCK <= '0';

SPI\_SS\_N <= '1';

DONE <= '1';

RDATA <= (others => '0');

state <= S0;

elsif rising\_edge( CLK ) then

case state is

when S0 =>

SPI\_SS\_N <= HOLD\_SS\_N; --initial status CS

SPI\_MOSI <= '0';

SPI\_SCK <= '0';

wait\_count <= WAIT\_COUNT\_MAX;

index\_send <= 1;

index\_resive <= 0; --add null bit

if START = '1' then -- start pulse detected

sreg <= WDATA; -- capture input data to send

DONE <= '0'; -- de-assert DONE

state <= S1; -- goto S1

end if;

when S1 =>

SPI\_SS\_N <= '0'; -- CS Low

SPI\_MOSI <= sreg(data\_M\_to\_S - index\_send); -- send data master to slave MSB first

SPI\_SCK <= '0'; -- SCK low

state <= S2;

when S2 =>

if wait\_count /= 0 then --wait\_count is not equal to 0,this loop wait until half SPI clk period

wait\_count <= wait\_count - 1; -- decrement wait counter

SPI\_SCK <= '0'; -- SCK low

else

SPI\_SCK <= '1'; -- SCK high

index\_send <= index\_send + 1; --increase index\_send

wait\_count <= WAIT\_COUNT\_MAX;

state <= S3;

end if;

when S3 =>

if wait\_count /= 0 then

wait\_count <= wait\_count - 1; -- decrement wait counter

else

SPI\_SCK <= '0'; -- SCK low

wait\_count <= WAIT\_COUNT\_MAX;

state <= S4;

end if;

when S4 => --check number of index data master send to slave

if index\_send > data\_M\_to\_S then

state <= S5;

else

state <= S1;

end if;

when S5 => --keep data from ADC

if wait\_count /= 0 then

wait\_count <= wait\_count - 1; -- decrement wait counter

else

SPI\_SCK <= '1'; -- SCK high

rreg(data\_S\_to\_M - index\_resive) <= SPI\_MISO; --keep data from ADC

wait\_count <= WAIT\_COUNT\_MAX;

state <= S6;

end if;

when S6 =>

if wait\_count /= 0 then

wait\_count <= wait\_count - 1; -- decrement wait counter

else

SPI\_SCK <= '0'; -- SCK high

index\_resive <= index\_resive + 1; --increase index\_resive

wait\_count <= WAIT\_COUNT\_MAX;

state <= S7;

end if;

when S7 =>

if index\_resive > data\_S\_to\_M then

state <= S8;

else

state <= S5;

end if;

when S8 =>

RDATA <= rreg(data\_S\_to\_M -1 downto 0);

SPI\_MOSI <= '1';

SPI\_SCK <= '0';

SPI\_SS\_N <= '1';

DONE <= '1';

state <= S0;

when others =>

state <= S0;

end case;

end if;

end process;

end behave;

**โค้ดส่วนการส่งข้อมูลกลับไปที่ Pc**

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.numeric\_std.all;

entity rs232\_tx is

generic(

SYSTEM\_SPEED : integer := 50e6; -- clock speed, in Hz

BAUDRATE : integer :=9600); -- baudrate

port(

clk\_i : in std\_logic; -- system clock

rst\_i : in std\_logic; -- synchronous reset, active-Low

req\_i : in std\_logic; -- Tx request

ack\_o : out std\_logic; -- Tx acknowledge

data\_i : in std\_logic\_vector(7 downto 0); -- Tx data

tx : out std\_logic ); -- Tx output

end rs232\_tx;

architecture behave of rs232\_tx is

constant MAX\_COUNTER: integer := (SYSTEM\_SPEED / BAUDRATE);

type state\_type is (

WAIT\_FOR\_REQ,

SEND\_START\_BIT,

SEND\_BITS,

SEND\_STOP\_BIT);

signal state: state\_type := WAIT\_FOR\_REQ;

signal baudrate\_counter: integer range 0 to MAX\_COUNTER := 0;

signal bit\_counter : integer range 0 to 7 := 0;

signal shift\_reg : std\_logic\_vector(7 downto 0) := (others => '0');

signal data\_sending\_started: std\_logic := '0';

begin

-- acknowledge, when sending process was started

--ack\_o <= data\_sending\_started and (not req\_i);

update: process(clk\_i)

begin

if rising\_edge(clk\_i) then

if rst\_i = '0' then

tx <= '1';

data\_sending\_started <= '0';

state <= WAIT\_FOR\_REQ;

else

case state is

-- wait until the master asserts valid data

when WAIT\_FOR\_REQ =>

ack\_o <= '0';

if req\_i = '0' then

state <= SEND\_START\_BIT;

baudrate\_counter <= MAX\_COUNTER - 1;

tx <= '0';

shift\_reg <= data\_i;

data\_sending\_started <= '1';

else

tx <= '1';

end if;

when SEND\_START\_BIT =>

if baudrate\_counter = 0 then

state <= SEND\_BITS;

baudrate\_counter <= MAX\_COUNTER - 1;

tx <= shift\_reg(0); -- send LSB first

bit\_counter <= 7;

else

baudrate\_counter <= baudrate\_counter - 1;

end if;

when SEND\_BITS =>

if baudrate\_counter = 0 then

if bit\_counter = 0 then

state <= SEND\_STOP\_BIT;

tx <= '1';

else

tx <= shift\_reg(1); -- send next bit

shift\_reg <= '0' & shift\_reg(7 downto 1);

bit\_counter <= bit\_counter - 1;

end if;

baudrate\_counter <= MAX\_COUNTER - 1;

else

baudrate\_counter <= baudrate\_counter - 1;

end if;

when SEND\_STOP\_BIT =>

if baudrate\_counter = 0 then

ack\_o <= '1';

state <= WAIT\_FOR\_REQ;

else

baudrate\_counter <= baudrate\_counter - 1;

end if;

end case;

-- this resets acknowledge until all bits are sent.

if req\_i = '1' and data\_sending\_started = '1' then

data\_sending\_started <= '0';

end if;

end if;

end if;

end process;

end behave;

------------------------------------------------------------------------------

**โค้ด Processing ส่วนหลัก**

import controlP5.\*;

import processing.serial.\*;

// main control p5

ControlP5 cp5;

// object of p5

Chart myChart, myChart2;

Textlabel status\_port, x1, x2, y1, y2, gx, gy;

Toggle ch1\_toggle, ch2\_toggle, stop\_toggle;

Button x\_ch1, y\_ch1, x\_ch2, y\_ch2, global\_x, global\_y, reconnect;

ColorWheel ch1\_color, ch2\_color;

// serial

Serial myPort;

String name; // store port name (last connect) for check port is available

// other var

int confirm\_connect = 0; // for ping board

boolean stop\_update = false; // for control update chart and read serial

int status\_reset = 0;

int x\_app = 0; // base value for draw gui

int y\_app = 0; // base value for draw gui

int mode\_ch = 0; //0 1 2 3

int max\_row = 10; // number of row

int max\_colum = 22; // number of colum

int multiple\_of\_row = 45; // distance between block in y axis

int multiple\_of\_colum = 50; // distance between block in x axis

int range\_y\_max = 6; // set max value for scale axis y

int buffer\_ch1 = 1100; // number of buffer ch1

int buffer\_ch2 = 1100; // number of buffer ch2

float scale\_y\_ch1 = 1; // voltage per block

float scale\_y\_ch2 = 1; // voltage per block

float scale\_x\_ch1 = 0.1; // time per block

float scale\_x\_ch2 = 0.1; // time per block

float send\_time = 0.00016; // time of data 0.00016 for 6000 package 0.005 for 200 package

int time\_out = 0; // for check port still available

// main colour 1 2 3 4 5 darker

color shade1 = color(255, 230, 250);

color background\_moniter = color(255,10);

color shade2 = color(196, 215, 237);

color shade3 = color(171, 200, 226);

color shade4 = color(55, 93, 129);

color shade5 = color(24, 49, 82);

void setup() {

size(1250, 850);

cp5 = new ControlP5(this);

ControlFont Arial = new ControlFont(createFont("Arial",15));

cp5.setFont(Arial);

// chart ch1

myChart = cp5.addChart("myChart")

.setPosition(x\_app+80, y\_app+80)

.setSize(1100, 450)

.setRange(-range\_y\_max, range\_y\_max)

.setView(Chart.LINE)

.setColorCaptionLabel(shade5)

.setColorBackground(background\_moniter)

.addDataSet("ch1")

.setData("ch1", new float[buffer\_ch1])

.setColors("ch1", color(0, 255 ,255))

;

//chart ch2

myChart2 = cp5.addChart("myChart2")

.setPosition(x\_app+80, y\_app+80)

.setSize(1100, 450)

.setRange(-range\_y\_max, range\_y\_max)

.setView(Chart.LINE)

.setColorCaptionLabel(shade5)

.setColorBackground(background\_moniter)

.addDataSet("ch2")

.setData("ch2", new float[buffer\_ch2])

.setColors("ch2", color(255, 0 ,0))

;

myChart.setStrokeWeight(2); // size of point in chart ch1

myChart2.setStrokeWeight(2); // size of point in chart ch2

// draw status port on or off

status\_port = cp5.addTextlabel("port")

.setText("PORT STATUS : OFF")

.setPosition(x\_app+1000, y\_app+50)

.setColorValue(0xffffff00)

.setFont(createFont("Arial",20))

;

// show scale x for ch1

x1 = cp5.addTextlabel("x1")

.setText(scale\_x\_ch1+" Sec")

.setPosition(x\_app+185, y\_app+660)

.setColorValue(0xffffff00)

.setFont(createFont("Arial",20))

;

// show scale x for ch2

x2 = cp5.addTextlabel("x2")

.setText(scale\_x\_ch2+" Sec")

.setPosition(x\_app+865, y\_app+660)

.setColorValue(0xffffff00)

.setFont(createFont("Arial",20))

;

// show scale y for ch1

y1 = cp5.addTextlabel("y1")

.setText(scale\_y\_ch1+" Volt")

.setPosition(x\_app+185, y\_app+730)

.setColorValue(0xffffff00)

.setFont(createFont("Arial",20))

;

// show scale y for ch2

y2 = cp5.addTextlabel("y2")

.setText(scale\_y\_ch2+" Volt")

.setPosition(x\_app+865, y\_app+730)

.setColorValue(0xffffff00)

.setFont(createFont("Arial",20))

;

// show number of colums

gx = cp5.addTextlabel("gx")

.setText("22 Box")

.setPosition(x\_app+600, y\_app+630)

.setColorValue(0xffffff00)

.setFont(createFont("Arial",20))

;

// show number of rows

gy = cp5.addTextlabel("gy")

.setText("10 Box")

.setPosition(x\_app+600, y\_app+740)

.setColorValue(0xffffff00)

.setFont(createFont("Arial",20))

;

// toggle for on / off ch1

ch1\_toggle = cp5.addToggle("ch1\_toggle")

.setPosition(x\_app+70, y\_app+570)

.setSize(100,40)

.setValue(false)

.setMode(ControlP5.SWITCH)

;

// toggle for on / off ch2

ch2\_toggle = cp5.addToggle("ch2\_toggle")

.setPosition(x\_app+750, y\_app+570)

.setSize(100,40)

.setValue(false)

.setMode(ControlP5.SWITCH)

;

// toggle for on / off update chart

stop\_toggle = cp5.addToggle("stop\_toggle")

.setBroadcast(false)

.setPosition(x\_app+100, y\_app+10)

.setSize(100,40)

.setValue(true)

.setMode(ControlP5.SWITCH)

.setBroadcast(true)

;

// button for change number of colum

global\_x = cp5.addButton("global\_x")

.setBroadcast(false)

.setPosition(x\_app+580, y\_app+570)

.setSize(100,50)

.setValue(0)

.setBroadcast(true)

;

// button for change number of row

global\_y = cp5.addButton("global\_y")

.setBroadcast(false)

.setPosition(x\_app+580, y\_app+680)

.setSize(100,50)

.setValue(0)

.setBroadcast(true)

;

// button for change x scale ch1

x\_ch1 = cp5.addButton("CH1\_X\_Scale")

.setBroadcast(false)

.setPosition(x\_app+70, y\_app+650)

.setSize(100,50)

.setValue(0)

.setBroadcast(true)

;

// button for change x scale ch2

x\_ch2 = cp5.addButton("CH2\_X\_Scale")

.setBroadcast(false)

.setPosition(x\_app+750, y\_app+650)

.setSize(100,50)

.setValue(0)

.setBroadcast(true)

;

// button for change y scale ch1

y\_ch1 = cp5.addButton("CH1\_Y\_Scale")

.setBroadcast(false)

.setPosition(x\_app+70, y\_app+720)

.setSize(100,50)

.setValue(0)

.setBroadcast(true)

;

// button for change y scale ch2

y\_ch2 = cp5.addButton("CH2\_Y\_Scale")

.setBroadcast(false)

.setPosition(x\_app+750, y\_app+720)

.setSize(100,50)

.setValue(0)

.setBroadcast(true)

;

// button for reconnect

reconnect = cp5.addButton("reconnect")

.setBroadcast(false)

.setPosition(x\_app+300, y\_app+10)

.setSize(100,50)

.setValue(0)

.setBroadcast(true)

;

// for set colour chart of ch1 , ch2

ch1\_color = cp5.addColorWheel("ch1\_color" , x\_app+300 , y\_app+560 , 200 ).setRGB(color(255,0,0));

ch2\_color = cp5.addColorWheel("ch2\_color" , x\_app+980 , y\_app+560 , 200 ).setRGB(color(0,255,0));

}

void draw() {

background(shade5);

drawBox();

serial\_check();

drawGrid();

}

**โค้ดส่วนควบคุม CH1**

// update x scale

void CH1\_X\_Scale(){

if(scale\_x\_ch1 == 0.01){

scale\_x\_ch1 = 0.025;

}else if(scale\_x\_ch1 == 0.025){

scale\_x\_ch1 = 0.05;

}else if(scale\_x\_ch1 == 0.05){

scale\_x\_ch1 = 0.1;

}else if(scale\_x\_ch1 == 0.1){

scale\_x\_ch1 = 0.2;

}else if(scale\_x\_ch1 == 0.2){

scale\_x\_ch1 = 0.00025;

}else if(scale\_x\_ch1 == 0.00025){

scale\_x\_ch1 = 0.0005;

}else if(scale\_x\_ch1 == 0.0005){

scale\_x\_ch1 = 0.001;

}else if(scale\_x\_ch1 == 0.001){

scale\_x\_ch1 = 0.01;

}

set\_x\_scale\_ch1();

}

// set new buffer size when change x scale

void set\_x\_scale\_ch1(){

buffer\_ch1 = int((scale\_x\_ch1 / send\_time)) \* max\_colum;

myChart.setData("ch1", new float[buffer\_ch1]);

x1.setText(scale\_x\_ch1+" sec");

}

// update y scale

void CH1\_Y\_Scale(){

if(scale\_y\_ch1 == 0.5){

scale\_y\_ch1 = 1;

}else if(scale\_y\_ch1 ==1){

scale\_y\_ch1 = 2;

}else if(scale\_y\_ch1 ==2){

scale\_y\_ch1 = 3;

}else if(scale\_y\_ch1 ==3){

scale\_y\_ch1 = 0.5;

}

y1.setText(scale\_y\_ch1+" Volt");

}

// update mode and read status

void ch1\_toggle(){

if(mode\_ch == 0){ // set on ch1

mode\_ch = 1;

buffer\_ch1 = int((scale\_x\_ch1 / send\_time)) \* max\_colum;

}else if(mode\_ch ==1){ // set off ch1

mode\_ch = 0;

buffer\_ch1 = 1;

}else if(mode\_ch ==2){ // set on ch1

mode\_ch = 3;

buffer\_ch1 = int((scale\_x\_ch1 / send\_time)) \* max\_colum;

}else if(mode\_ch ==3){ // set off ch1

mode\_ch = 2;

buffer\_ch1 = 1;

}

myChart.setData("ch1", new float[buffer\_ch1]);

send\_command\_set\_mode();

}

// update colour

void ch1\_color(){

int red\_ch1 = cp5.get(ColorWheel.class,"ch1\_color").r();

int green\_ch1 = cp5.get(ColorWheel.class,"ch1\_color").g();

int blue\_ch1 = cp5.get(ColorWheel.class,"ch1\_color").b();

myChart.setColors("ch1", color(red\_ch1,green\_ch1,blue\_ch1));

}

**โค้ดส่วนควบคุม CH2**

// update x scale

void CH2\_X\_Scale(){

if(scale\_x\_ch2 == 0.01){

scale\_x\_ch2 = 0.025;

}else if(scale\_x\_ch2 == 0.025){

scale\_x\_ch2 = 0.05;

}else if(scale\_x\_ch2 == 0.05){

scale\_x\_ch2 = 0.1;

}else if(scale\_x\_ch2 == 0.1){

scale\_x\_ch2 = 0.2;

}else if(scale\_x\_ch2 == 0.2){

scale\_x\_ch2 = 0.00025;

}else if(scale\_x\_ch2 == 0.00025){

scale\_x\_ch2 = 0.0005;

}else if(scale\_x\_ch2 == 0.0005){

scale\_x\_ch2 = 0.001;

}else if(scale\_x\_ch2 == 0.001){

scale\_x\_ch2 = 0.01;

}

set\_x\_scale\_ch2();

}

// set new buffer size when change x scale

void set\_x\_scale\_ch2(){

buffer\_ch2 = int((scale\_x\_ch2 / send\_time)) \* max\_colum;

myChart2.setData("ch2", new float[buffer\_ch2]);

x2.setText(scale\_x\_ch2+" sec");

}

// update y scale

void CH2\_Y\_Scale(){

if(scale\_y\_ch2 == 0.5){

scale\_y\_ch2 = 1;

}else if(scale\_y\_ch2 ==1){

scale\_y\_ch2 = 2;

}else if(scale\_y\_ch2 ==2){

scale\_y\_ch2 = 3;

}else if(scale\_y\_ch2 ==3){

scale\_y\_ch2 = 0.5;

}

y2.setText(scale\_y\_ch2+" Volt");

}

// update mode and read status

void ch2\_toggle(){

if(mode\_ch == 0){ // on ch2

mode\_ch = 2;

buffer\_ch2 = int((scale\_x\_ch2 / send\_time)) \* max\_colum;

}else if(mode\_ch ==1){ // on ch2

buffer\_ch2 = int((scale\_x\_ch2 /send\_time)) \* max\_colum;

mode\_ch = 3;

}else if(mode\_ch == 2){ // off ch2

mode\_ch = 0;

buffer\_ch2 = 1;

}else if(mode\_ch == 3){ // off ch2

mode\_ch = 1;

buffer\_ch2 = 1;

}

myChart2.setData("ch2", new float[buffer\_ch2]);

send\_command\_set\_mode();

}

// update colour

void ch2\_color(){

int red\_ch2 = cp5.get(ColorWheel.class,"ch2\_color").r();

int green\_ch2 = cp5.get(ColorWheel.class,"ch2\_color").g();

int blue\_ch2 = cp5.get(ColorWheel.class,"ch2\_color").b();

myChart2.setColors("ch2", color(red\_ch2,green\_ch2,blue\_ch2));

}

**โค้ดส่วนควบคุม ตาราง**

// re to base value

void resetChart(){

if(mode\_ch ==1){ // set off ch2

buffer\_ch1 = 100;

set\_x\_scale\_ch1();

}else if(mode\_ch ==2){ // set off ch 1

buffer\_ch2 = 100;

set\_x\_scale\_ch2();

}else if(mode\_ch ==3){ // set on all

buffer\_ch1 = 100;

buffer\_ch2 = 100;

set\_x\_scale\_ch1();

set\_x\_scale\_ch2();

}

}

//set number colum

void global\_x(){

if(max\_colum < 44){

max\_colum \*= 2;

}else{

max\_colum = 11;

}

multiple\_of\_colum = 1100/max\_colum;

gx.setText(max\_colum+" Box");

set\_x\_scale\_ch1();

set\_x\_scale\_ch2();

}

// set number row

void global\_y(){

if(max\_row == 10){

max\_row = 14;

}else if(max\_row == 14){

max\_row = 6;

}else{

max\_row = 10;

}

multiple\_of\_row = 450/max\_row;

gy.setText(max\_row+" Box");

}

// stop update or re update

void stop\_toggle(){

if(stop\_update){

stop\_update = false;

resetChart();

}else{

stop\_update = true;

}

}

// call reconnect function

void reconnect(){

check\_serial\_again();

}

**ส่วนวาดรูปตารางและกรอบ**

// draw box options

void drawBox(){

stroke(shade3);

fill(shade4);

rect(x\_app+50, y\_app+550, 490 ,240); // ch1 control

rect(x\_app+720, y\_app+550, 490 ,240); // ch2 control

rect(x\_app+80, y\_app+5, 140, 70); // stop toggle

rect(x\_app+280, y\_app+5, 140, 70); // reconnect

fill(shade4);

rect(x\_app+550, y\_app+550, 160, 240); // global control

fill(shade5);

stroke(shade1);

rect(x\_app+175, y\_app+650, 120, 50); // ch1 show x scale

rect(x\_app+175, y\_app+720, 120, 50); // ch1 show y scale

rect(x\_app+580, y\_app+625, 100, 40); // show number colum

rect(x\_app+580, y\_app+735, 100, 40); // show number row

rect(x\_app+855, y\_app+650, 120, 50); // ch2 show x scale

rect(x\_app+855, y\_app+720, 120, 50); // ch2 show y scale

fill(255);

rect(x\_app+80, y\_app+80, 1100, 450); // chart background

}

// draw colum and row in chart

void drawGrid(){

stroke(shade2);

for(int colum = 0 ; colum < max\_colum ; colum ++){

line(x\_app+80+(colum\*multiple\_of\_colum), y\_app+80, x\_app+80+(colum\*multiple\_of\_colum), y\_app+529);

}

for(int row = 0 ; row <= max\_row ; row ++){

line(x\_app+80 , y\_app+80+(row\*multiple\_of\_row), x\_app+1180, y\_app+80+(row\*multiple\_of\_row));

}

stroke(0);

line(x\_app+80, y\_app+305, x\_app+80+1100, y\_app+305); // zero line

}

**ส่วนการรับค่าและนำมาแสดงผล รวมทั้งคำนวน**

void serial\_check(){

if (myPort != null ){

status\_port.setText("PORT STATUS : ON");

// for check serial port is available it look like when port come name = 80 then if port left port = 81

// 80 != 81 it will not reset value thus it will let pc know port is not available

try {

if(!(name == Serial.list()[0])){

time\_out = 0;

}

}catch (Exception e) {

print(" port miss ");

}

//

if(stop\_update == false){ // update chart allow ?

if(myPort.available() > get\_delay()){ // delay for store update value

while(myPort.available() > 5){ // 5 because in worth case read command want 5 package for update

int in\_mode = myPort.read();

if((in\_mode) != (mode\_ch+128)){ // mode input match with pc mode or not

send\_command\_set\_mode();

confirm\_connect = 0;

myPort.clear(); // reset input form port

}else{

// true value is value1+value2 in string mode like value 1 = 20 value 2 = 23

// true value is ''20''+''23'' = ''2023'' mV then before send to calculate divide by 1000

// for set in volt , 2.023 volt

int value\_first;

int value\_second;

int value\_first2;

int value\_second2;

switch(in\_mode){

case 128: // mode 0

break;

case 129: // mode 1

value\_first = myPort.read();

value\_second = myPort.read();

myChart.push("ch1", calculate\_output\_with\_scale(float((value\_first\*100)+(value\_second))/1000, scale\_y\_ch1));

break;

case 130: // mode 2

value\_first = myPort.read();

value\_second = myPort.read();

myChart2.push("ch2", calculate\_output\_with\_scale(float((value\_first\*100)+(value\_second))/1000, scale\_y\_ch2));

break;

case 131: // mode 3

value\_first = myPort.read();

value\_second = myPort.read();

value\_first2 = myPort.read();

value\_second2 = myPort.read();

myChart.push("ch1", calculate\_output\_with\_scale(float((value\_first\*100)+(value\_second))/1000, scale\_y\_ch1));

myChart2.push("ch2", calculate\_output\_with\_scale(float((value\_first2\*100)+(value\_second2))/1000, scale\_y\_ch2));

break;

}

}

}

}

}

// ping board for let board know we are here

confirm\_connect++;

if(confirm\_connect == 10){

confirm\_connect = 0;

send\_command\_set\_mode();

}

}else{

status\_port.setText("PORT STATUS : OFF");

}

// if time out that mean usb is not available it will reset port

if(time\_out >= 100){

myPort = null;

status\_port.setText("PORT STATUS : OFF");

time\_out = 0;

}

time\_out++;

}

// calculate output for map in chart

float calculate\_output\_with\_scale(float input, float scale){

float output = 2/float(max\_row);

output \*= (range\_y\_max\*input)/scale;

return output;

}

// set mode to board

void send\_command\_set\_mode(){

myPort.write('#');

myPort.write(str(mode\_ch));

}

// try to reconnect if port available

void check\_serial\_again(){

if(Serial.list() != null){

name = Serial.list()[0];

myPort = new Serial(this, Serial.list()[0], 460800);

delay(50);

}

}

// for delay update by highest scale(mean lowest buffer)

float get\_delay(){

float higest\_time;

if( scale\_x\_ch1 < scale\_x\_ch2){

higest\_time = scale\_x\_ch1;

}else{

higest\_time = scale\_x\_ch2;

}

if(higest\_time >= 0.1){

return (0);

}

if(mode\_ch == 3){

return (1000);

}

return (600); // size\_of\_data \* 3 \* 0.1

}