Name: _____ ID: _____ Start Date: Thursday, August 12, 2021

Name: _____ ID: _____ Due Date: Friday, August 13, 2021

Software and Hardware Co-Design with Zybo, Summer 2021 HUST
Lab #2 Phase 2
Reading Three TMP101 Sensors and UART with Zybo Board

This is a group lab to be performed by groups of two students. Each group would need to submit one report that should include names and ID numbers of both students. All submitted simulation waveforms and oscilloscope waveform photos or screen captures must be annotated, initialed, and dated by both students of the group.

1. Deliverables Due by Friday, August 13, 2021

- *Demonstrate your TMP101-based thermometer on your Zybo board to display the address and temperature from one of two TMP101 sensors in Fahrenheit and Celsius on a serial terminal on your laptop. The temperature is displayed in two decimal digits from 0 to 99 degrees.*
- *The lower 2-bit address of the TMP101 sensor to be displayed is set by two slide switches. There are three available addresses: 2'b00, 2'b01, 2'b10 but only two of them are used. When the 2-bit address changes, its corresponding TMP101 sensor should be sensed and its 2-digit temperature in both Fahrenheit and Celsius should be displayed on the serial terminal automatically.*
- *Submit a pdf copy of simulation waveforms for your controller. The simulation waveforms must be signed by all members of the group.*
- *Submit a hard copy of the I2C bus waveforms to send addresses and read TMP101 chips with at least two different addresses captured on an oscilloscope. Show screen shots of complete I2C frame transmissions. Annotate your waveforms to show temperature values. The waveforms must be initialed and dated by both students.*

2. Available Verilog Source Modules from the Instructor

ClockedPositiveOneShot.v, ClockedNegativeOneShot.v SendToUart.v, OneTemperatureConverter.v, uart_BaudRateGenerator.v, uart_tx.v, bbfifo_16x8.v, kcuart_tx.v.

The following test benches are available: ControllerReadTempI2C_tb.v, I2C_ShiftRegister_tb.v, uart_BaudRateGenerator_tb.v.

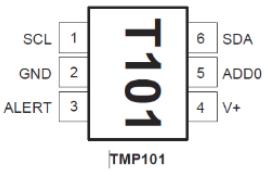
Two template files are to be used to develop your controller and top-level circuit. ControllerReadTMP101template.v and lab2phase2TMP101summer2021template.v. ControllerReadTMP101TB.v is the test bench to verify your controller.

3. Objectives

To read the upper 8-bits of the 12-bit temperature registers from a TMP101 chip with one of three different addresses, one at a time and display the temperature of the chip currently addressed in Fahrenheit and Celsius in two decimal digits for each TMP101 on a serial terminal connected to a Zybo board.

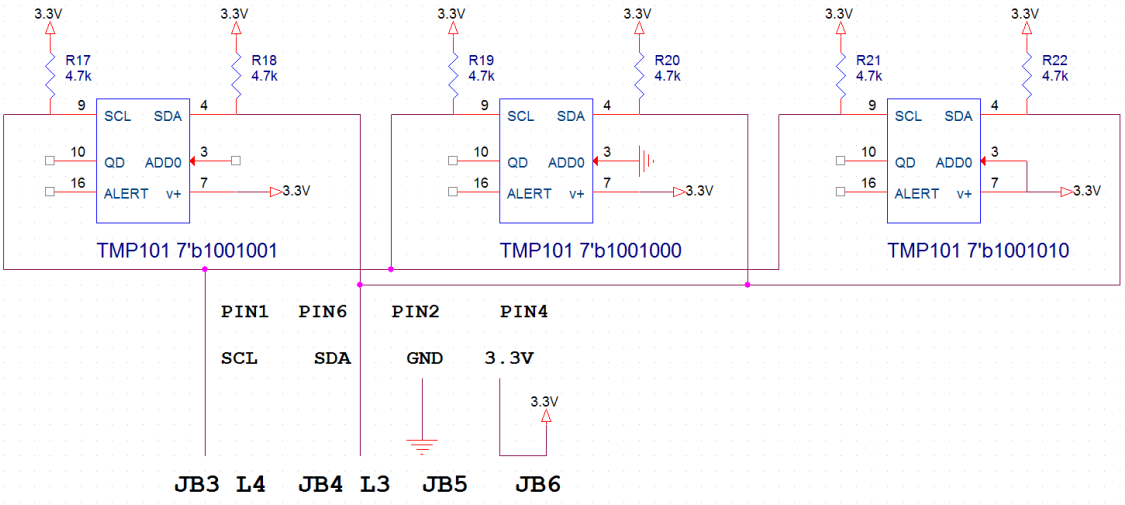
ControllerReadTMP101.v will need to be created based on the controller from Phase I to read temperature from TMP101 thermometers.

4. TMP101 Interface



TMP101 is a 6-pin SOT23 device. SOT23 stands for small outline transistor 23. It is about 2mm x 3mm. The chip is soldered on to a 6-pin TMP101 PCB board provided in the lab kit. It can be connected to one of Pmod connectors of the Zybo board.

A 4-pin TMP101 board adaptor shown below is made on a perforated board to make it easy to connect the TMP101 board to a Pmod connector of Zybo board. Three possible addresses are possible. The default is address 7'b1001001 when AD0 is floating. The other two addresses are 7'b1001000 when AD0 is grounded and 7'b1001010 when AD0 is at 3.3V. TMP101 can be connected to Zybo as shown in the following schematic. One possible connection is JB3=SCL, JB4=SDA, JB5=GND, and JB6=3.3.



5. TMP101 breakout Boards and I2C waveforms on Zybo Board

5.1 Pin Assignment on Zybo Board

The following pins are suggested but you are free to use other pins except for 125MHz clock on pin L16. Mode=1 for continuous reading. Press Start on BTN0 to start.

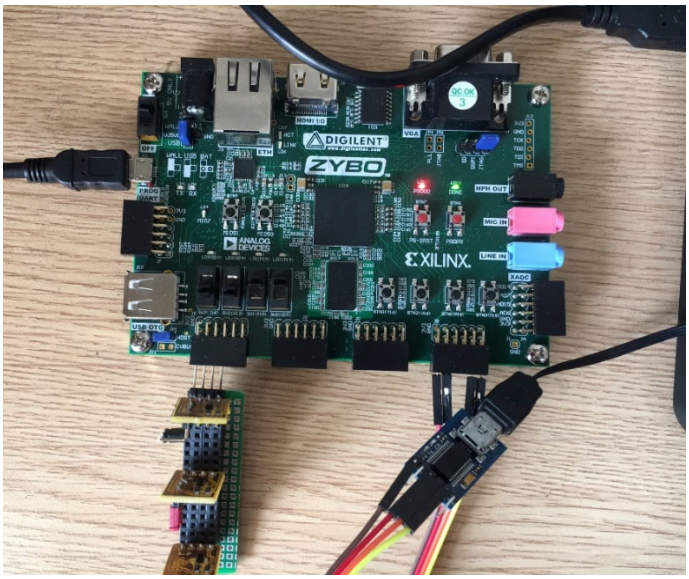
5.2 A photo of three TMP101 sensors on a Zybo board

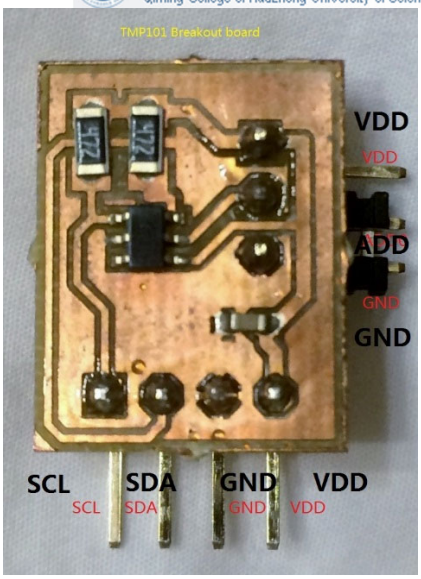
Temperature is displayed through a serial port connector on Pmod JB connector. Notice three TMP101 breakout boards are connected on the same I2C bus line on Pmod JE connector. I2C address is set with SW3-1. Start button is BTN0 and Mode button is SW0.

Project Summary | Lab2I2Cphase2TMP101fall2020JJS.v | SendTemperature.v

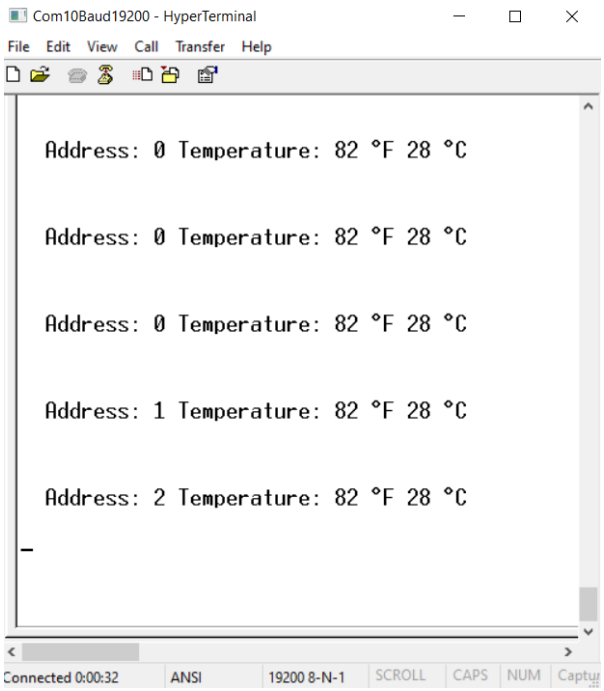
C:/SHCodesign2020vivado/lab2fall2020HUST_I2C/Lab2I2Cphase2TMP101fall2020JJS/Lab2I2Cp

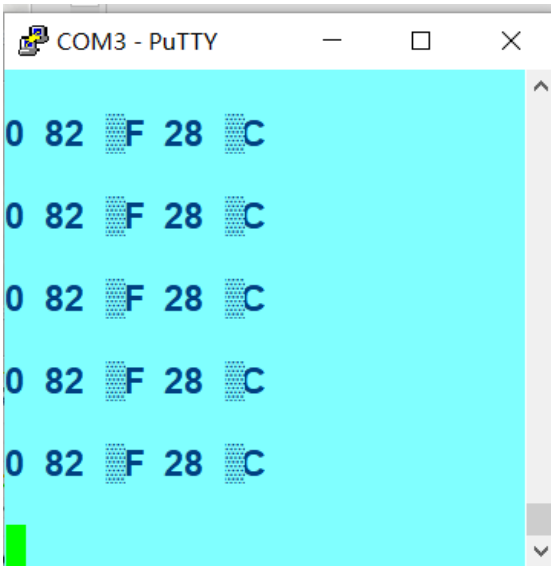
```
1  #125 MHz oscillator
2  set_property PACKAGE_PIN L16 [get_ports Clock]
3  #LD0
4  set_property PACKAGE_PIN M14 [get_ports ClockLocked]
5  #btn0
6  set_property PACKAGE_PIN R18 [get_ports Reset]
7  #JD4
8  set_property PACKAGE_PIN R14 [get_ports SDA]
9  #JD3
10 set_property PACKAGE_PIN P14 [get_ports SCL]
11 #JB1
12 set_property PACKAGE_PIN T20 [get_ports tx]
13 #JB2
14 set_property PACKAGE_PIN U20 [get_ports rx]
15 #BTN3
16 set_property PACKAGE_PIN Y16 [get_ports Start]
17 #SW3
18 set_property PACKAGE_PIN T16 [get_ports Mode]
19 #SW1 and SW2
20 set_property PACKAGE_PIN W13 [get_ports {Address[1]}]
21 set_property PACKAGE_PIN P15 [get_ports {Address[0]}]
```





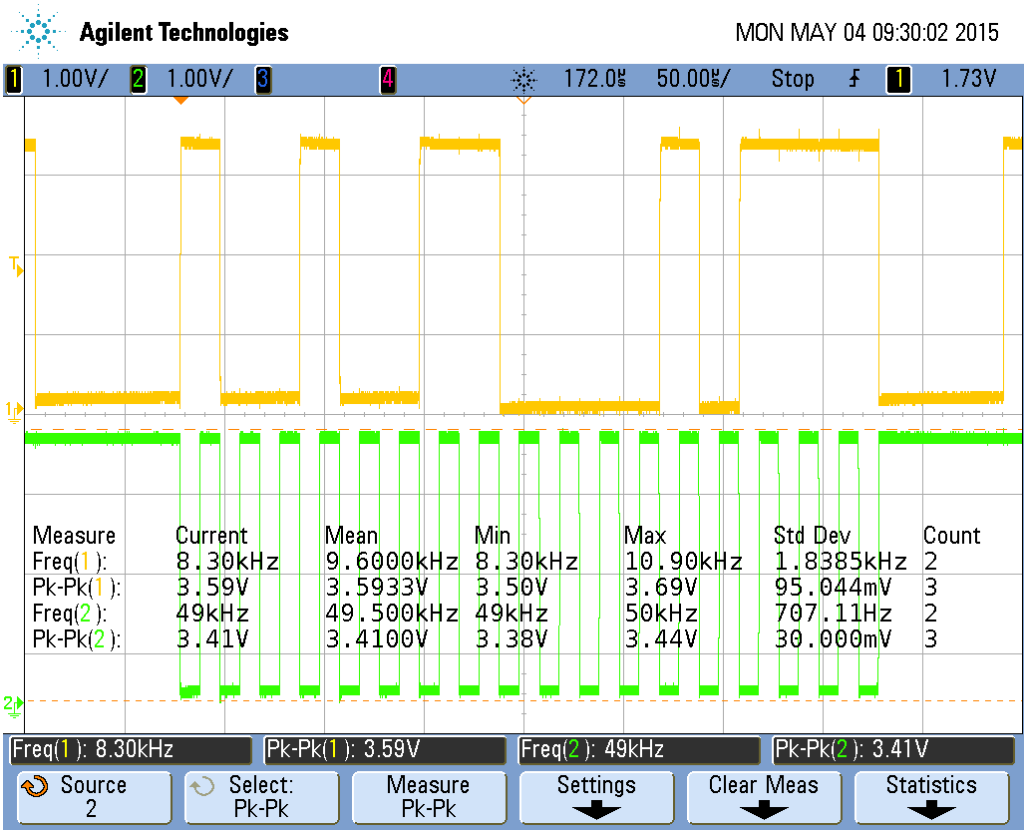
Temperature is displayed in both Fahrenheit and Celsius after the address of a TMP101 is displayed as Addresses 0, 1, or 2 as shown below that is set on SW2 and SW1. For a laptop that runs a non-English Windows, the special character for degree ° will not be displayed correctly as shown on the right-hand side. See the appendix to this hand for a solution.





5.3 Single TMP101 snap shot waveforms

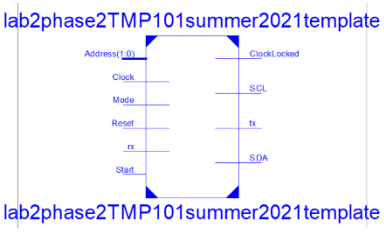
First byte is 8'b1001001. Second byte is 8'b00010111=23 °C.



6. Reading One TMP101 Sensor and Display Temperatures in Fahrenheit and Celsius

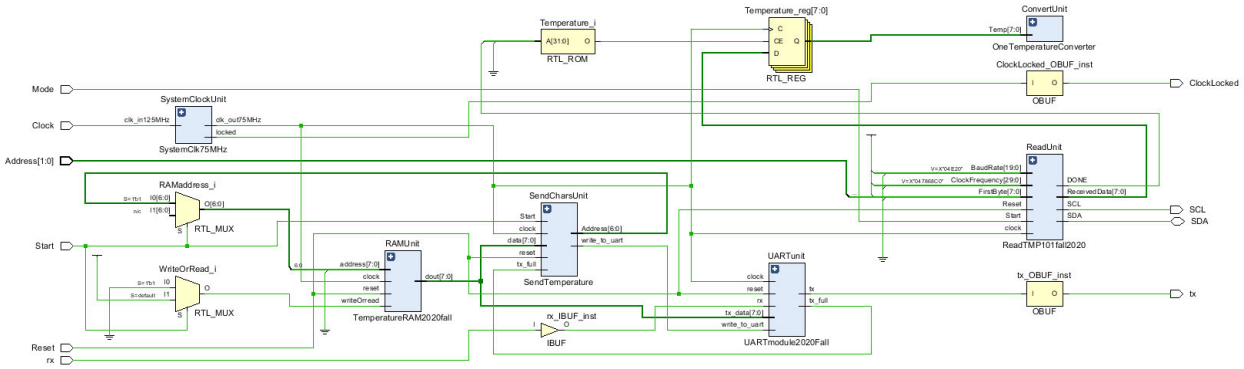
Module lab2phase2TMP101summer2021template.v is the top level module for this lab. Address is a 3-bit number to select the lower 3-bits of a TMP101 sensor. ChipSelect is connected to three slide switches. The Mode is also connected to a slide switch to choose a single-shot mode or continuous reading mode. On the single shot mode, push button Start is pressed to start a single reading of temperature. One continuous mode, the circuit will read temperature of a TMP101 continuously.

6.1 Module lab2ph2TMP101summer2021template.v



Top-level template module lab2phase2TMP101summer2021template.v is available from the instructor as shown on the right hand side. Address is a 2-bit input to set the lower 2-bit address of a TMP101. Start is a push-button signal to start sending temperature display to the serial port. Two digit temperature is displayed in both Fahrenheit and Celsius through UART transmit module to a serial terminal on a laptop. DONE==1 when the reading is complete, which is not connected to an external display.

This module is composed of a number of submodules: ReadTempI2C2017spring, OneTemperatureConverster, send to UART etc. ReadTempI2C2017spring.v is the main submodule that needs to be created and debugged.



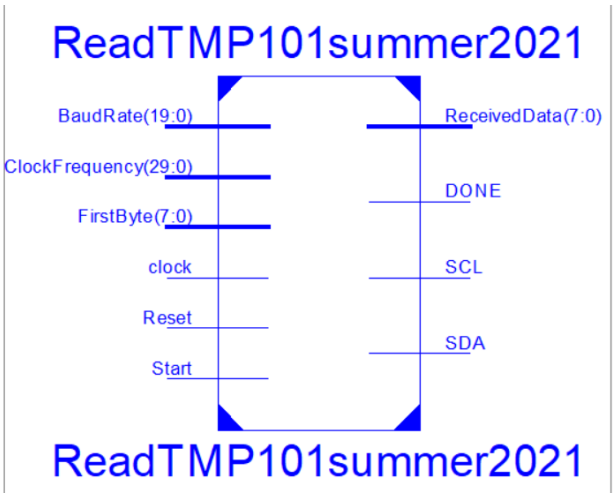

```
//File Name: lab2phase2TMP101summer2021template.v
//Author: Jianjian Song
//Date: November 2021
//HUST Summer 2021
//This is a template as a starting point for students
//Address is from  two slide switches: SW2 and SW1 typing
//Read selected TMP101 temperature sensor
//send first address and Read byte to I2C bus with address {4'b1001,ChipSelect}
//Receive first byte temperature from the selected TMP101 on I2C bus
//Display the current address and its temperature to a uart display
//Display 2-digit temperature in Fahrenheit and 2-digit in Celsius
//Mode = 0 for single read mode when Start is pressed to sample once
//Mode = 1 for continuous sampling mode

module lab2phase2TMP101summer2021template (input Reset, Clock, rx, Mode, Start,
input [1:0] Address,
output tx, SCL, ClockLocked, inout SDA);

parameter I2CBaudRate=20'd15000, ClockFrequency=30'd60000000;
parameter RAMAddressSize=7;
 wire clock60MHz;
SystemClk60MHz SystemClockUnit(.clk_in125MHz(Clock),.clk_out60MHz(clock60MHz),
.LOCKED(ClockLocked));

wire [7:0] Chip;
wire WriteLoad, ReadorWrite, ShiftorHold, Select, BaudEnable, StartStopAck;

//module ReadTMP101summer2021(input Reset, clock, Start,
//input [7:0] FirstByte,
//input [19:0] BaudRate,
//input [29:0] ClockFrequency,
//output DONE, SCL,
//inout SDA,
wire [7:0] ReceivedData;
//select an address to read
assign Chip = {5'b10010,Address,1'b1};
ReadTMP101summer2021 ReadUnit(Reset, clock60MHz, StartReading||Mode, Chip, I2CBaudRate,
ClockFrequency,
DONE, SCL, SDA, ReceivedData);

wire [3:0] Fahtenth, Fahones, Celtenth, Celones;
reg [7:0] Temperature;
always@(posedge clock60MHz)
   if(DONE==1) Temperature<=ReceivedData;
   else Temperature<=Temperature;

//odule OneTemperatureConverter( input [7:0] Temp,
//output [3:0] Fahtenth, Fahones, Celtenth, Celones);

OneTemperatureConverter ConvertUnit(Temperature, Fahtenth, Fahones, Celtenth, Celones);
```

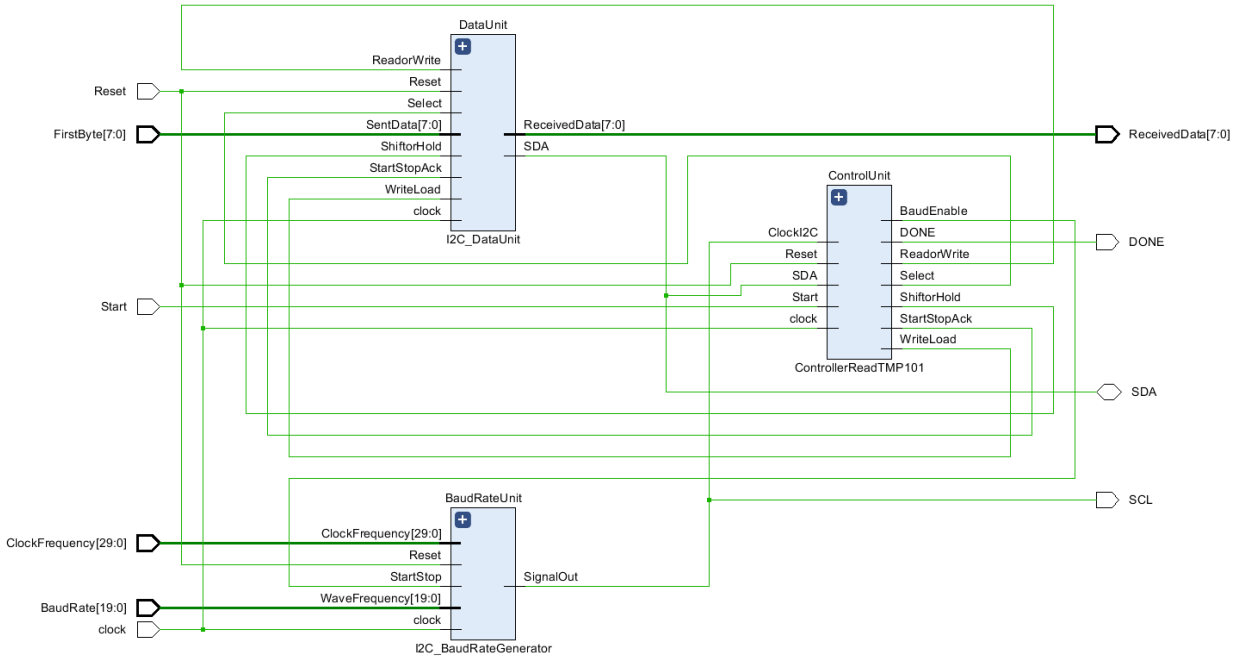
```
reg [RAMAddressSize-1:0] WriteAddress;
wire [RAMAddressSize-1:0] ReadAdress, RAMaddress;

wire WriteorRead;

reg [7:0] DataIn;
wire [7:0] RAMDataout;

assign RAMaddress = (Start==1)?ReadAdress:WriteAddress;
assign   WriteOrRead=(Start==1)?0:1;

//module UARTmodule2021summer #(parameter TRANSMITTED_BITS=8, BAUDRATE=20'd19200,
FREQUENCY=30'd125000000)
//(input  reset, clock, read_from_uart, write_to_uart, rx,
//input [TRANSMITTED_BITS-1:0] tx_data,
//output  tx_full, rx_data_present, tx,
//output [TRANSMITTED_BITS-1:0] rx_data,
//output reg [TRANSMITTED_BITS-1:0] transmitted_bits);
// Signals for UART connections

wire read_from_uart, write_to_uart;
wire          tx_full;
wire          tx_half_full;
wire          rx_data_present;
wire          rx_full;
wire          rx_half_full;
wire [7:0] rx_data, tx_data;

//module TemperatureRAM2021summer #(parameter DataWidth=7, MemorySize=100, AddressBits=7)
//(input reset, clock, writeOrread,
//input [AddressBits:0] address,
//input [DataWidth-1:0] din,
//output [DataWidth-1:0] dout);

TemperatureRAM2021summer RAMUnit(Reset, clock60MHz, WriteOrRead, RAMaddress, DataIn, RAMDataout);

//module SendTemperature #(parameter AddressBits=7)(
//input Start, tx_full, reset, clock, input [6:0] data,
//output reg write_to_uart, Transmitting,
//output reg [AddressBits-1:0] Address);

assign tx_data=RAMDataout;

SendTemperature SendCharsUnit(Start, tx_full, Reset, clock60MHz, RAMDataout, write_to_uart, Transmitting,
ReadAdress);

//module UARTmodule2021summer #(parameter TRANSMITTED_BITS=8, BAUDRATE=20'd19200,
FREQUENCY=30'd60000000)
//(input  reset, clock, read_from_uart, write_to_uart, rx,
//input [TRANSMITTED_BITS-1:0] tx_data,
//output  tx_full, rx_data_present, tx,
//output [TRANSMITTED_BITS-1:0] rx_data,
//output reg [TRANSMITTED_BITS-1:0] transmitted_bits);

UARTmodule2021summer UARTunit(Reset, clock60MHz, read_from_uart, write_to_uart, rx, tx_data, tx_full,
rx_data_present,
tx, rx_data, );

endmodule
```

6.2 Module ReadTMP101summer2021.v

This module will start reading a TMP101 from address FirstByte when input Start=1 and display 8-bit temperature as ReceviedData and set Done =1 to indicate completion of

temperature reading. I2C baud rate and system clock frequency are also to inputs to this module.





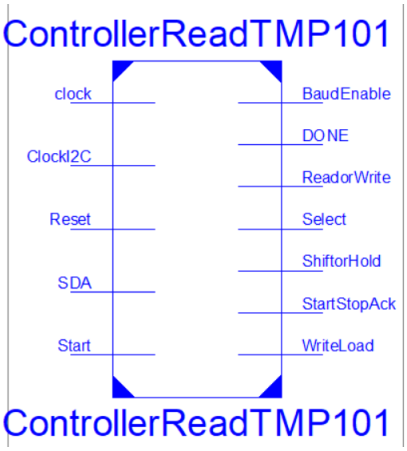
This module is composed of three submodules: I2C_DataUnit.v and I2C_BaudRateGenerator.v and ControllerReadTMP101.v. The first two modules are created from Phase I of this lab. ControllerReadTMP101.v will need to be created.

6.3 ControllerReadTMO101template.v

The controller uses one shots from both positive and negative edges of SCL signal to synchronize the state changes and counter updates. The example circuit statements for them are given below as well as a recommended ASM chart for the controller.

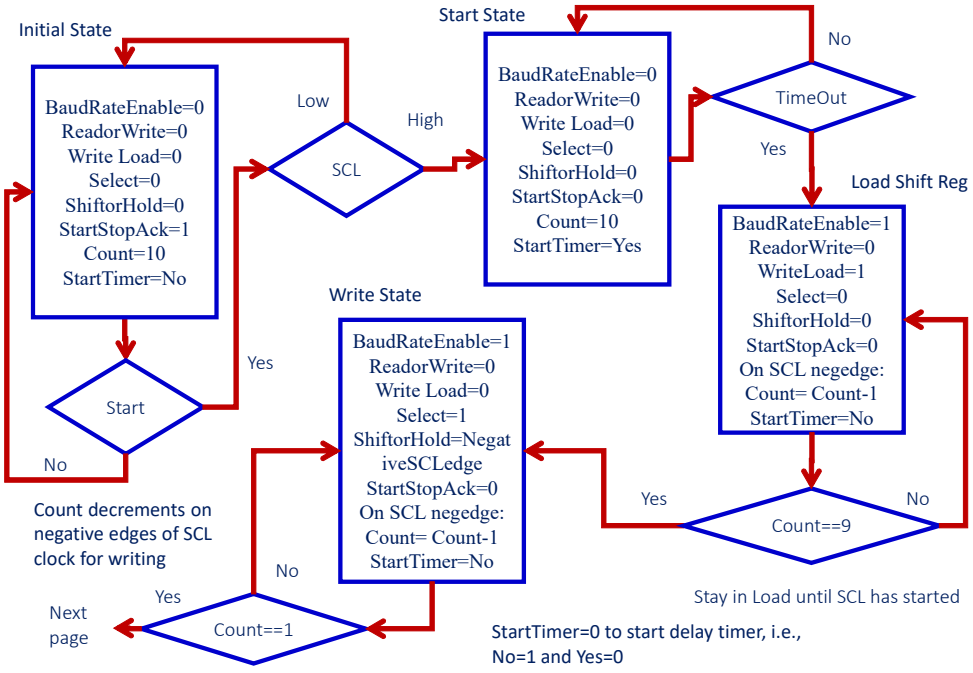
```
wire OneShotNegative, OneShotPositive;
ClockedNegativeOneShot OneShotNegativeUnit(ClockI2C, OneShotNegative, Reset, clock);
ClockedPositiveOneShot OneShotPositiveUnit(ClockI2C, OneShotPositive, Reset, clock);
reg ACKbit;
always@(posedge clock)
   if(Reset==1) begin State<=InitialState; ACKbit<=1; end
```

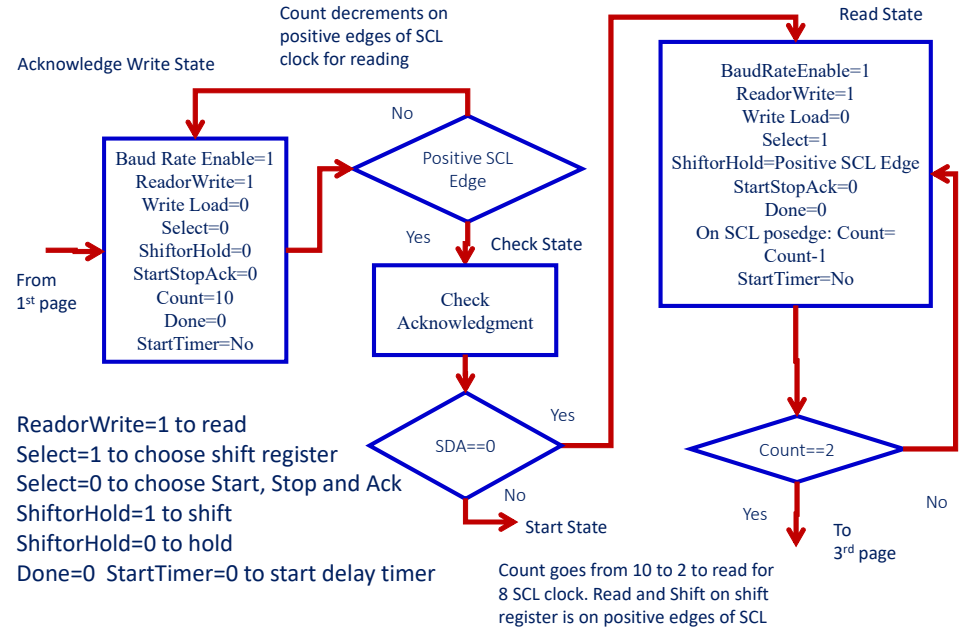
```
   else begin State<=NextState;
                if(OneShotPositive==1) ACKbit<=SDA; else ACKbit<=ACKbit; end
//count update
always@(posedge clock)
if(Reset==1) begin DataCounter<=4'd10; end
else
case (State)
LoadState: if(OneShotNegative==0) DataCounter<=DataCounter-1'b1; else
DataCounter<=DataCounter;
WriteState: if(OneShotNegative==0) DataCounter<=DataCounter-1'b1; else
DataCounter<=DataCounter;
ReadState: if(OneShotPositive==1) DataCounter<=DataCounter-1'b1; else
DataCounter<=DataCounter;
default: DataCounter<=4'd10;
endcase
```

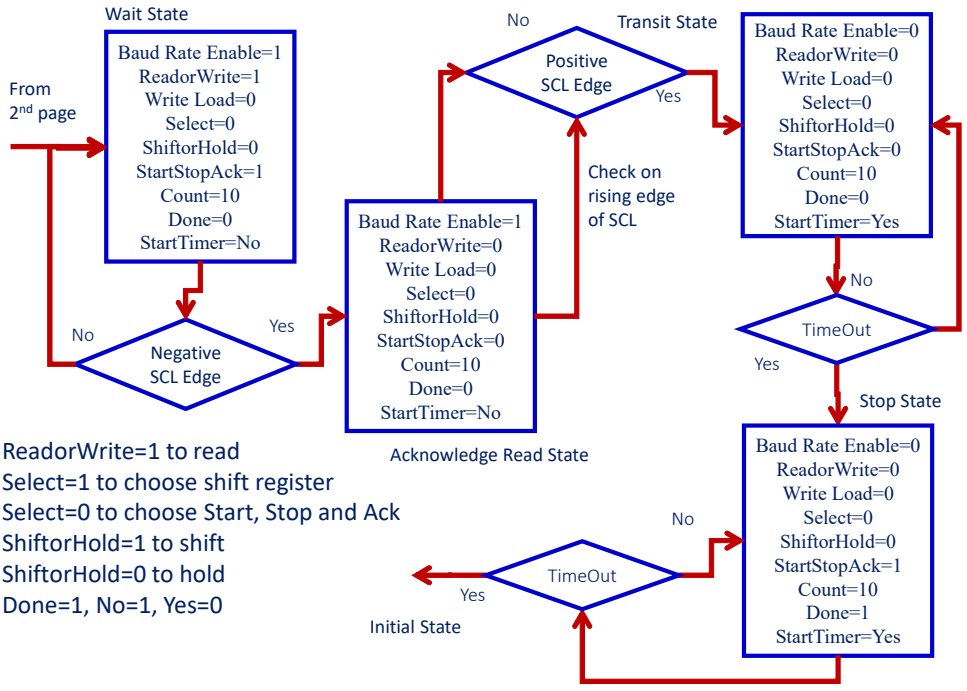


Here is a template for the new controller.

```
module ControllerReadTMP101template (input Reset, clock, Start, ClockI2C, SDA,
output reg WriteLoad, ReadorWrite, ShiftorHold, Select, BaudEnable,
output reg StartStopAck, DONE);

parameter InitialState=4'd0, StartState=4'd1, LoadState=4'd2, WriteState=4'd3, AcknowledgeState1=4'd4;
parameter ConnectedState=4'd5, ReadState=4'd6, WaitState=4'd7, AcknowledgeState2=4'd8;
parameter TransitState=4'd9, StopState=4'd10;

reg [3:0] DataCounter;
reg [3:0] State, NextState;

wire TimeOut;
reg StartDelayLoop;
//module DelayTimeReset(input Reset, Clock, Start, output reg Timeout);
DelayTimeReset DelayUnit(Reset, clock, StartDelayLoop,TimeOut);

wire OneShotNegative, OneShotPositive;
ClockedNegativeOneShot OneShotNegativeUnit(ClockI2C, OneShotNegative, Reset, clock);
ClockedPositiveOneShot OneShotPositiveUnit(ClockI2C, OneShotPositive, Reset, clock);

reg ACKbit;
always@(posedge clock) begin
   if(Reset==1) begin State<=InitialState; ACKbit<=1; end
   else begin State<=NextState;
   if(OneShotPositive==1) ACKbit<=SDA; else ACKbit<=ACKbit; end
end

endmodule
```





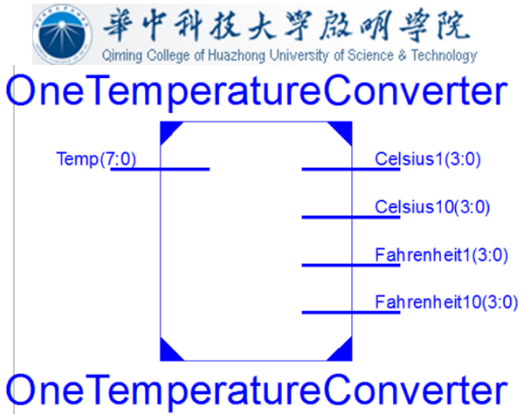


6.4 Temperature Conversion

The temperature conversion circuits are given below. The first circuit converts one temperature to both Fahrenheit and Celsius and the second circuit converts two temperatures to Fahrenheit.

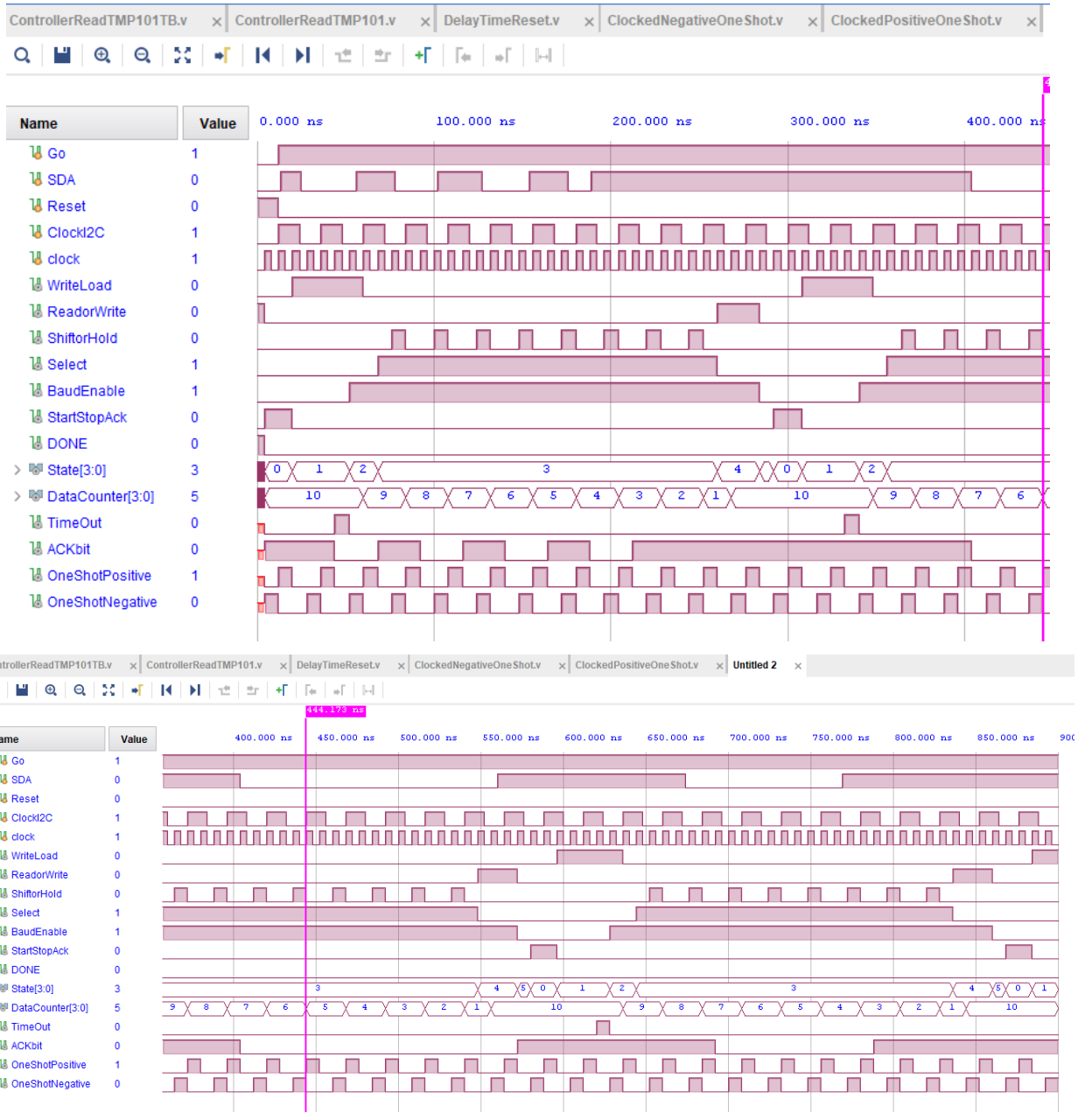
```
module OneTemperatureConverter(Temp, Fahrenheit10, Fahrenheit1, Celsius10, Celsius1);
```

```
input [7:0] Temp;
output [3:0] Fahrenheit10, Fahrenheit1, Celsius10, Celsius1;

wire [14:0] longtemp;

assign longtemp = (Temp*5'd18/4'd10+6'd32);

assign Fahrenheit10 = longtemp/4'd10;
assign Fahrenheit1 = longtemp%4'd10;

assign Celsius10 = Temp/4'd10;
assign Celsius1 = Temp%4'd10;

endmodule
```



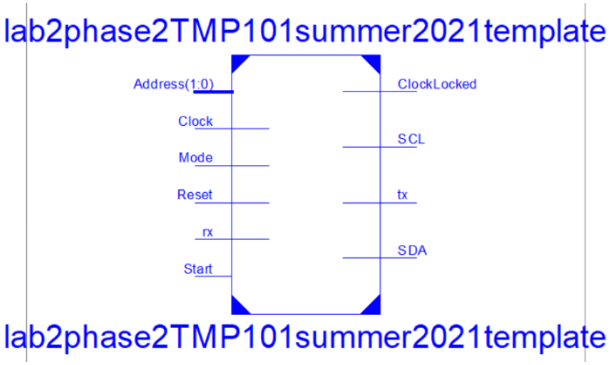
6.5 ControllerReadTMP101.v Simulation

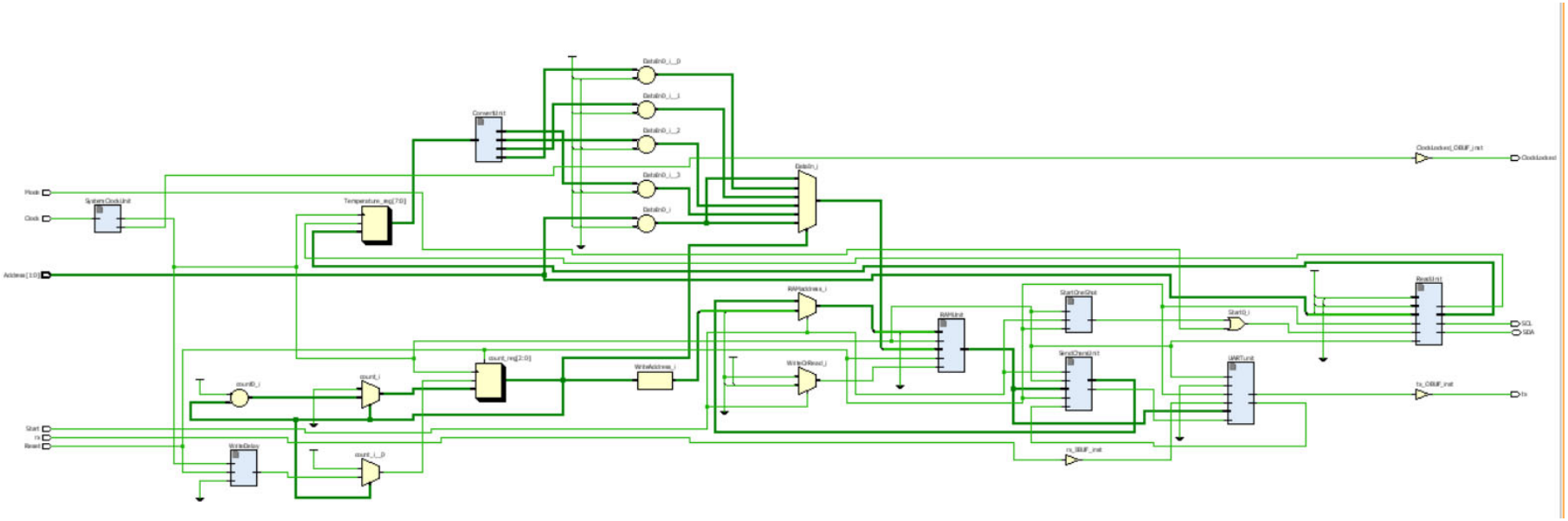
Here is one simulation outcome of the controller.



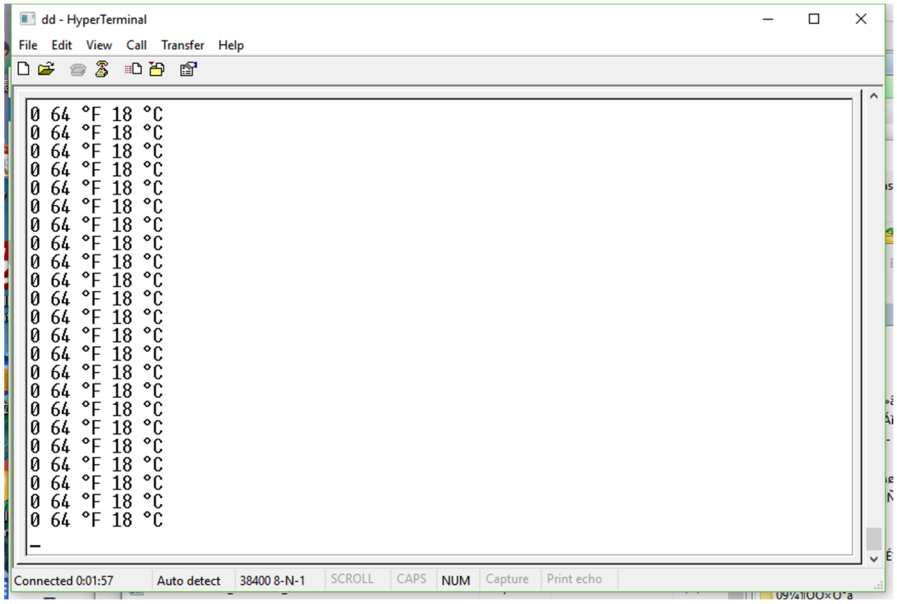
This is the test bench that generates above waveforms.

```
`timescale 1ns / 1ps
//set MaxCount =3 for simulation in module DelayTime()
//Test bench for Phase 2 controller
//HUST 2020 Fall
//Lab #2 Phase 2
//parameter  MaxCount = 3;
module ControllerReadTMP101TB;
   reg Go, SDA, Reset, ClockI2C, clock;

   wire WriteLoad, ReadorWrite, ShiftorHold, Select, BaudEnable, StartStopAck,DONE;
   wire [3:0] State=uut.State;
   wire [3:0] DataCounter=uut.DataCounter;
   wire TimeOut=uut.TimeOut, ACKbit=uut.ACKbit;
   wire OneShotPositive=uut.OneShotPositive, OneShotNegative=uut.OneShotNegative;

//module ControllerReadTMP101 (input Reset, clock, Start, ClockI2C, SDA,
//output reg WriteLoad, ReadorWrite, ShiftorHold, Select, BaudEnable,
//output reg StartStopAck, DONE);
//
ControllerReadTMP101 uut (Reset, clock, Go, ClockI2C, SDA, WriteLoad, ReadorWrite, ShiftorHold, Select,
   BaudEnable, StartStopAck, DONE);

   initial begin  Go = 0;  Reset = 0;  ClockI2C = 0;  clock = 0; end
   always #4 clock=~clock;
   always #12 ClockI2C=~ClockI2C;
   initial fork
   #0 Go = 0;  #12 Go = 1;
   #0 Reset = 1;  #12 Reset = 0;
   #0 SDA=0;  #13 SDA=1; #25 SDA=0; #56 SDA=1; #78 SDA=0; #102 SDA=1;
   #127 SDA=0; #154 SDA=1; #176 SDA=0; #189 SDA=1;
   #404 SDA=0;         #560 SDA=1;  #674 SDA=0; #769 SDA=1; #404 SDA=0;  #821 SDA=1;
   #900 $stop;
   join

endmodule
```

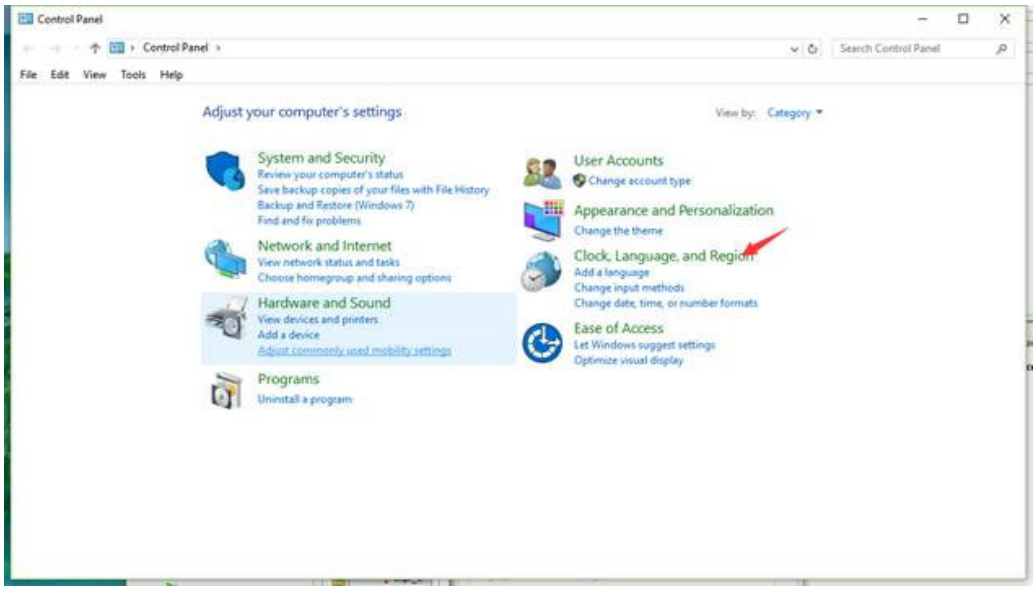

lab2phase2TMP101summer2021template
Address(1:0)
Clock
Mode
Reset
rx
Start
ClockLocked
SCL
tx
SDA
lab2phase2TMP101summer2021template

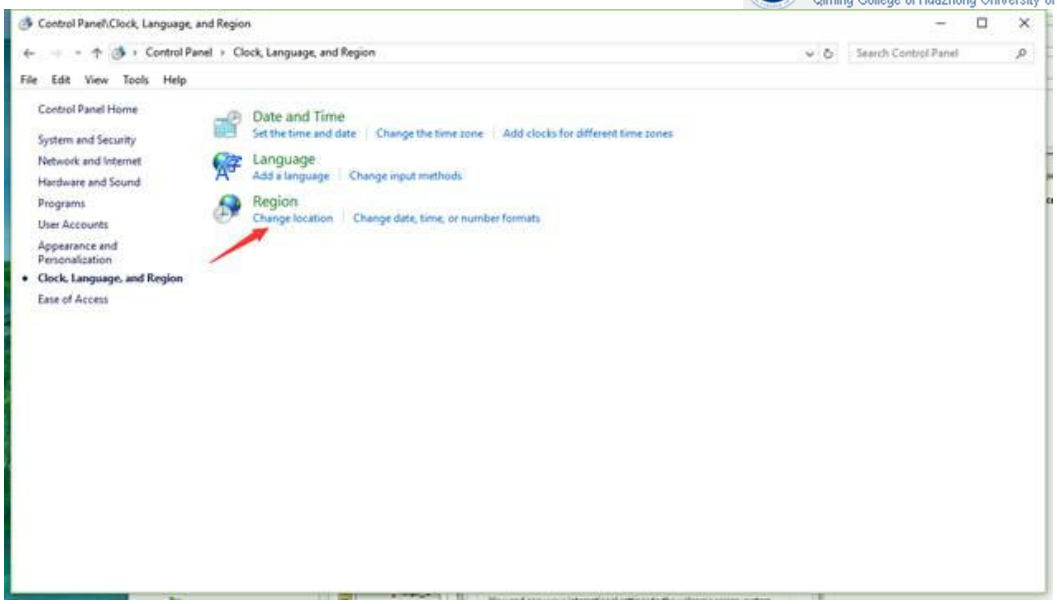


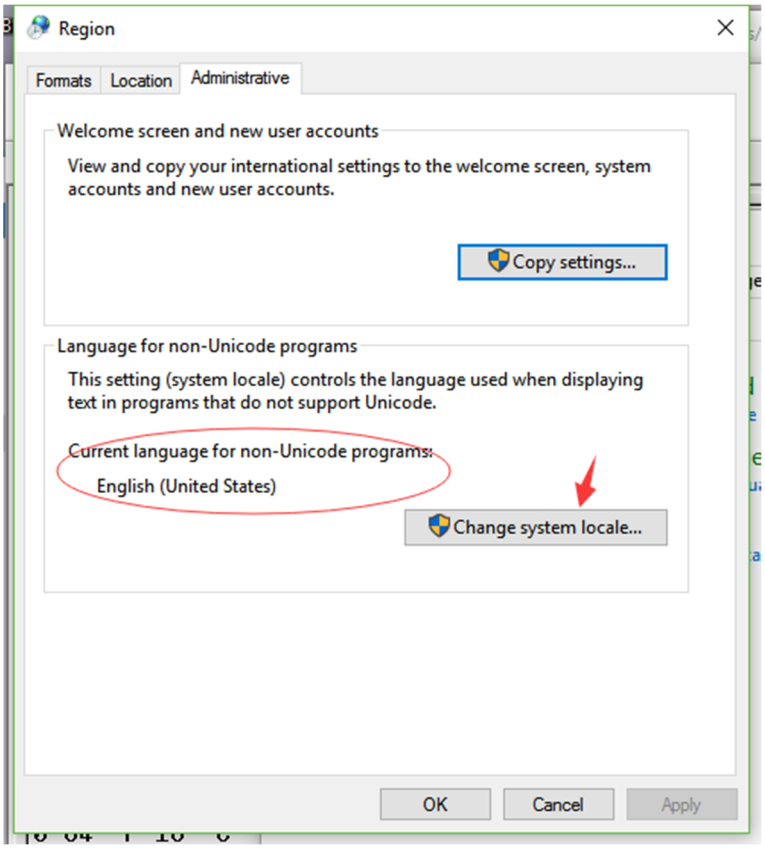
7. Appendix: A solution from Huang Zhen to Display 8-bit Special ASCII Characters



The reason is the system encoding/decoding settings on Windows. For Chinese Simplified edition, the language for non-Unicode programs is set as default Chinese, but English edition is not. So, we can fix it by changing system locale like this.



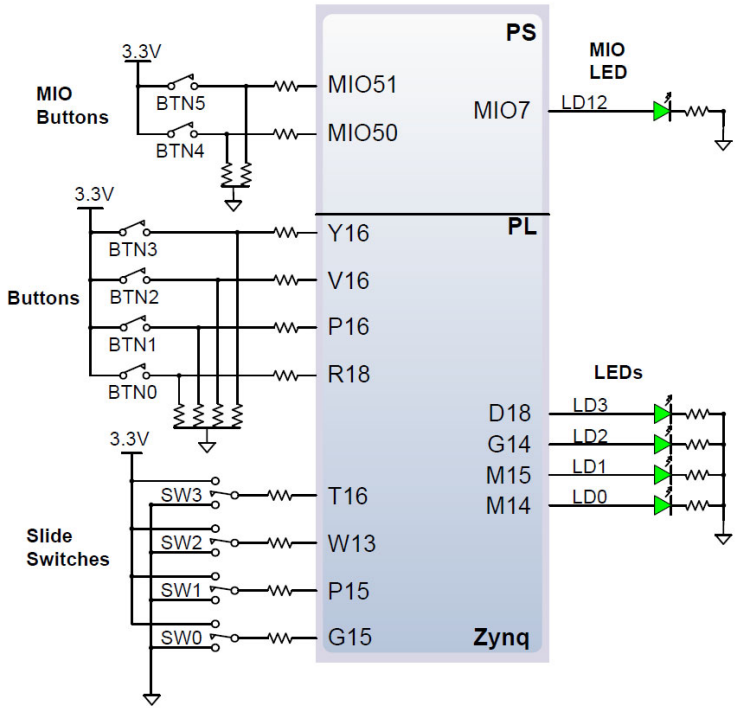




Best regards.

黄 振
Danceiny / Spring Oscillator
Email：danceiny@gmail.com
GitHub：github.com/danceiny
Group：Dian/Meigoo/Iris, 14th SeedClass of HUST

8. Peripherals and Pmod Connectors on the Zybo Board



Pmod JA (XADC)	Pmod JB (Hi-Speed)	Pmod JC (Hi-Speed)	Pmod JD (Hi-Speed)	Pmod JE (Std.)	Pmod JF (MIO)
JA1: N15	JB1: T20	JC1: V15	JD1: T14	JE1: V12	JF1: MIO-13
JA2: L14	JB2: U20	JC2: W15	JD2: T15	JE2: W16	JF2: MIO-10
JA3: K16	JB3: V20	JC3: T11	JD3: P14	JE3: J15	JF3: MIO-11
JA4: K14	JB4: W20	JC4: T10	JD4: R14	JE4: H15	JF4: MIO-12
JA7: N16	JB7: Y18	JC7: W14	JD7: U14	JE7: V13	JF7: MIO-0
JA8: L15	JB8: Y19	JC8: Y14	JD8: U15	JE8: U17	JF8: MIO-9
JA9: J16	JB9: W18	JC9: T12	JD9: V17	JE9: T17	JF9: MIO-14
JA10: J14	JB10: W19	JC10: U12	JD10: V18	JE10: Y17	JF10: MIO-15

Table 9. Pmod pinout.

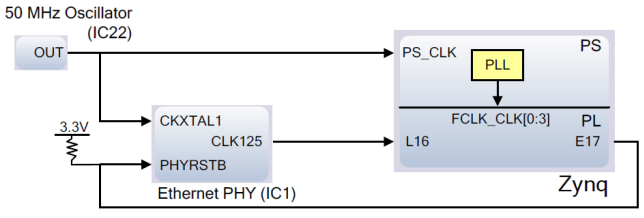


Figure 13. ZYBO clocking.

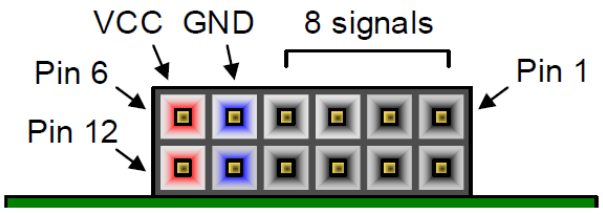


Figure 16. Pmod diagram.