**Date Submitted: 10/12/19**

**Task 00: Execute provided code**

**Youtube Link:** <https://www.youtube.com/watch?v=ZMlKZXVSnrM&list=PLLbVEP8QAFUFUZ1Mjoa5-TEAfeUglZqTq&index=2&t=0s>

**------------------------------------------------------------------------------------**

**Task 01:**

Youtube Link: <https://www.youtube.com/watch?v=ybIQIe6qFBM&list=PLLbVEP8QAFUFUZ1Mjoa5-TEAfeUglZqTq&index=2>

**Modified Schematic (if applicable):**

**N/A**

**Modified Code:**

**// Insert code here**

#include <stdint.h>

#include <stdlib.h>

#include <stdio.h>

#include <stdarg.h>

#include <stdbool.h>

#include "inc/hw\_memmap.h"

#include "inc/hw\_types.h"

#include "driverlib/gpio.h"

#include "driverlib/pin\_map.h"

#include "driverlib/sysctl.h"

#include "driverlib/uart.h"

#include "inc/hw\_ints.h"

#include "driverlib/interrupt.h"

//includes/defines from lab5 for temperature analysis

#include "driverlib/debug.h"

#include "inc/tm4c123gh6pm.h"

#include "driverlib/adc.h"

#include "driverlib/rom.h"

#include "driverlib/timer.h"

//Global variables to track temperature gauges

uint32\_t ui32ADC0Value[4];

uint32\_t ui32Period;

volatile uint32\_t ui32TempAvg;

volatile uint32\_t ui32TempValueC;

volatile uint32\_t ui32TempValueF, ui32TempValueF2;

int main(void)

{

//General settings

SysCtlClockSet(SYSCTL\_SYSDIV\_5|SYSCTL\_USE\_PLL|SYSCTL\_OSC\_MAIN|SYSCTL\_XTAL\_16MHZ); //changed the SYSCTL\_SYSDIV from 4 to 5

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_UART0);

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOA);

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOF);

GPIOPinTypeUART(GPIO\_PORTA\_BASE, GPIO\_PIN\_0 | GPIO\_PIN\_1);

GPIOPinConfigure(GPIO\_PA0\_U0RX);

GPIOPinConfigure(GPIO\_PA1\_U0TX);

//Initialize GPIO

GPIOPinTypeGPIOOutput(GPIO\_PORTF\_BASE, GPIO\_PIN\_1 | GPIO\_PIN\_2 | GPIO\_PIN\_3); //added GPIO\_PIN's 1 and 3 for task1

UARTConfigSetExpClk(UART0\_BASE, SysCtlClockGet(), 115200, (UART\_CONFIG\_WLEN\_8 | UART\_CONFIG\_STOP\_ONE | UART\_CONFIG\_PAR\_NONE));

//GPIO settings

GPIOPinConfigure(GPIO\_PA0\_U0RX);

GPIOPinConfigure(GPIO\_PA1\_U0TX);

GPIOPinTypeGPIOOutput(GPIO\_PORTF\_BASE, GPIO\_PIN\_1 | GPIO\_PIN\_2 | GPIO\_PIN\_3);

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

// Included initializations for Task1

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

//ADC settings

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_ADC0);

ROM\_ADCHardwareOversampleConfigure(ADC0\_BASE, 32);

ROM\_ADCSequenceConfigure(ADC0\_BASE, 1, ADC\_TRIGGER\_PROCESSOR, 0);

ROM\_ADCSequenceStepConfigure(ADC0\_BASE, 1, 0, ADC\_CTL\_TS);

ROM\_ADCSequenceStepConfigure(ADC0\_BASE, 1, 1, ADC\_CTL\_TS);

ROM\_ADCSequenceStepConfigure(ADC0\_BASE, 1, 2, ADC\_CTL\_TS);

ROM\_ADCSequenceStepConfigure(ADC0\_BASE,1,3,ADC\_CTL\_TS|ADC\_CTL\_IE|ADC\_CTL\_END);

ROM\_ADCSequenceEnable(ADC0\_BASE, 1);

//Timer 1 settings

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_TIMER1); //enable peripheral timer1

TimerConfigure(TIMER1\_BASE, TIMER\_CFG\_PERIODIC); //configure timer1 to periodic mode

ui32Period = (SysCtlClockGet()/2);

IntEnable(INT\_TIMER1A);

TimerLoadSet(TIMER1\_BASE, TIMER\_A, ui32Period -1);

TimerIntEnable(TIMER1\_BASE, TIMER\_TIMA\_TIMEOUT);//enables a specific event within the timer to generate an interrupt

IntMasterEnable();//master interrupt enable API for all interrupts

TimerEnable(TIMER1\_BASE, TIMER\_A);//enable the timer

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

while (1); //main loop

}

void Timer1IntHandler(void)

{

//Clears the timer1 interrupt

TimerIntClear(TIMER1\_BASE, TIMER\_TIMA\_TIMEOUT);

ADCIntClear(ADC0\_BASE, 1); //clears ADC

ADCProcessorTrigger(ADC0\_BASE, 1);

while(!ADCIntStatus(ADC0\_BASE, 1, false));

ADCSequenceDataGet(ADC0\_BASE, 1, ui32ADC0Value);

//ADC Calculations

ui32TempAvg = (ui32ADC0Value[0] + ui32ADC0Value[1] + ui32ADC0Value[2] + ui32ADC0Value[3] + 2)/4;

ui32TempValueC = (1475 - ((2475 \* ui32TempAvg)) / 4096)/10;

ui32TempValueF = ((ui32TempValueC \* 9) + 160) / 5;

//Convert ADC values to char variables to print to terminal

ui32TempValueF2 = ui32TempValueF;

volatile uint32\_t ones, tens;

ones = (ui32TempValueF2%10) + '0';

tens = ui32TempValueF2/10 + '0';

UARTCharPut(UART0\_BASE, tens);

UARTCharPut(UART0\_BASE, ones);

//Labels the printed ADC values as fahrenheit (F)

UARTCharPut(UART0\_BASE, 'F');

UARTCharPut(UART0\_BASE, '\n');

UARTCharPut(UART0\_BASE, '\r');

}

**------------------------------------------------------------------------------------**

**Task 02:**

Youtube Link: <https://www.youtube.com/watch?v=Bo6kfM88crM&list=PLLbVEP8QAFUFUZ1Mjoa5-TEAfeUglZqTq&index=3>

**Modified Schematic (if applicable):**

**N/A**

**Modified Code:**

#include <stdint.h>

#include <stdbool.h>

#include "inc/hw\_memmap.h"

#include "inc/hw\_types.h"

#include "driverlib/gpio.h"

#include "driverlib/pin\_map.h"

#include "driverlib/sysctl.h"

#include "driverlib/uart.h"

#include "inc/hw\_ints.h"

#include "driverlib/interrupt.h"

#include "driverlib/adc.h"

#include <string.h>

#include <math.h>

#include <stdlib.h>

#include "driverlib/rom.h"

uint32\_t ui32ADC0Value[4];

uint32\_t ui32Status;

volatile uint32\_t ui32TempAvg;

volatile uint32\_t ui32TempValueC;

volatile uint32\_t ui32TempValueF, ui32TempValueF2;

int main(void)

{

//General settings

SysCtlClockSet(SYSCTL\_SYSDIV\_5|SYSCTL\_USE\_PLL|SYSCTL\_OSC\_MAIN|SYSCTL\_XTAL\_16MHZ); //changed the SYSCTL\_SYSDIV from 4 to 5

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_UART0);

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOA);

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOF); //temp gpio

//Initialize GPIO and UART stuff

GPIOPinConfigure(GPIO\_PA0\_U0RX);

GPIOPinConfigure(GPIO\_PA1\_U0TX);

GPIOPinTypeUART(GPIO\_PORTA\_BASE, GPIO\_PIN\_0 | GPIO\_PIN\_1);

GPIOPinTypeGPIOOutput(GPIO\_PORTF\_BASE, GPIO\_PIN\_1 | GPIO\_PIN\_2 | GPIO\_PIN\_3); //added GPIO\_PIN's 1 and 3 for task1

UARTConfigSetExpClk(UART0\_BASE, SysCtlClockGet(), 115200, (UART\_CONFIG\_WLEN\_8 | UART\_CONFIG\_STOP\_ONE | UART\_CONFIG\_PAR\_NONE));

IntMasterEnable();//master interrupt enable API for all interrupts

IntEnable(INT\_UART0);

UARTIntEnable(UART0\_BASE, UART\_INT\_RX | UART\_INT\_RT);

//ADC settings

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_ADC0);

ROM\_ADCHardwareOversampleConfigure(ADC0\_BASE, 32);

ROM\_ADCSequenceConfigure(ADC0\_BASE, 1, ADC\_TRIGGER\_PROCESSOR, 0);

ROM\_ADCSequenceStepConfigure(ADC0\_BASE, 1, 0, ADC\_CTL\_TS);

ROM\_ADCSequenceStepConfigure(ADC0\_BASE, 1, 1, ADC\_CTL\_TS);

ROM\_ADCSequenceStepConfigure(ADC0\_BASE, 1, 2, ADC\_CTL\_TS);

ROM\_ADCSequenceStepConfigure(ADC0\_BASE,1,3,ADC\_CTL\_TS|ADC\_CTL\_IE|ADC\_CTL\_END);

ROM\_ADCSequenceEnable(ADC0\_BASE, 1);

//ROM\_ADCIntEnable(UART0\_BASE, UART\_INT\_RX | UART\_INT\_RX);

//ROM\_ADCIntEnable(ADC0\_BASE, 1);

while (1); //main loop

}

void UARTIntHandler(void)

{

ui32Status = UARTIntStatus(UART0\_BASE, true); //interrupt status

char newchar; //character value for cmd

UARTIntClear(UART0\_BASE, ui32Status);

while(UARTCharsAvail(UART0\_BASE)) //checks which character in terminal

{

newchar = UARTCharGet(UART0\_BASE); //set newchar equal to the character typed by the user

UARTCharPut(UART0\_BASE, newchar); //echo the character typed by the user

if(newchar == 'R')

{

GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_2, 0x00); //turn LED off to switch colors

GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_3, 0x00); //turn LED off to switch colors

GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_1, 0x02); //red LED on

}

if(newchar == 'r')

GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_1, 0x00); //red LED off

if(newchar == 'G')

{

GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_1, 0x00); //turn LED off to switch colors

GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_2, 0x00); //turn LED off to switch colors

GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_3, 0x08); //Green LED on

}

if(newchar == 'g')

GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_3, 0x00); //Green LED off

if(newchar == 'B')

{

GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_1, 0x00); //turn LED off to switch colors

GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_3, 0x00); //turn LED off to switch colors

GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_2, GPIO\_PIN\_2); //Blue LED on

}

if(newchar == 'b')

GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_2, 0x00); //blue LED off

if(newchar == 'T')

{

ROM\_ADCIntClear(ADC0\_BASE, 1);

ROM\_ADCProcessorTrigger(ADC0\_BASE, 1);

while(!ROM\_ADCIntStatus(ADC0\_BASE, 1, false));

ROM\_ADCSequenceDataGet(ADC0\_BASE, 1, ui32ADC0Value);

UARTCharPut(UART0\_BASE, ':');

UARTCharPut(UART0\_BASE, ' ');

//ADC Calculations

ui32TempAvg = (ui32ADC0Value[0] + ui32ADC0Value[1] + ui32ADC0Value[2] + ui32ADC0Value[3] + 2)/4;

ui32TempValueC = (1475 - ((2475 \* ui32TempAvg)) / 4096)/10;

ui32TempValueF = ((ui32TempValueC \* 9) + 160) / 5;

//Convert ADC values to char variables to print to terminal

ui32TempValueF2 = ui32TempValueF;

volatile uint32\_t ones, tens;

ones = (ui32TempValueF2%10) + '0';

tens = ui32TempValueF2/10 + '0';

UARTCharPut(UART0\_BASE, tens);

UARTCharPut(UART0\_BASE, ones);

UARTCharPut(UART0\_BASE, 'F'); //print F for fahrenheit

UARTCharPut(UART0\_BASE, '\n'); //print new line

UARTCharPut(UART0\_BASE, '\r'); //return on new line

}

}

}