**Date Submitted: 10/07/18**

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

**Task 01:**

Youtube Link: https://www.youtube.com/watch?v=cv1rzv1TBmU

**Modified Code:**

**// Insert code here**

#include <stdint.h>

#include <stdbool.h>

#include "inc/hw\_memmap.h"

#include "inc/hw\_types.h"

#include "driverlib/gpio.h"

#include "driverlib/interrupt.h"

#include "driverlib/pin\_map.h"

#include "driverlib/sysctl.h"

#include "driverlib/adc.h"

#include "driverlib/rom.h"

#include "driverlib/timer.h"

#include "inc/tm4c123gh6pm.h"

// header files that enable UART interrupts

#include "driverlib/uart.h"

// header file to manipulate characters

#include <ctype.h>

int main(void) {

/\* Variables for ADC \*/

volatile uint32\_t ui32TempAvg;

volatile uint32\_t ui32TempValueC;

volatile uint32\_t ui32TempValueF;

/\* CPU Clock SETUP () \*/

SysCtlClockSet(SYSCTL\_SYSDIV\_4 | SYSCTL\_USE\_PLL | SYSCTL\_OSC\_MAIN | SYSCTL\_XTAL\_16MHZ);

/\* UART SETUP \*/

// enable UART0 & Port A peripherals

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_UART0);

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOA);

// configure Tx and Rx pins as UART pins

GPIOPinConfigure(GPIO\_PA0\_U0RX);

GPIOPinConfigure(GPIO\_PA1\_U0TX);

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

// LED SETUP as part of UART

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOF); // enable GPIO port for LED

GPIOPinTypeGPIOOutput(GPIO\_PORTF\_BASE, GPIO\_PIN\_2); // enable pin for LED PF2

// initialize UART parameters: 115200 Baud Rate, 8 data bits, 1 stop bit

UARTConfigSetExpClk(UART0\_BASE, SysCtlClockGet(), 115200,

(UART\_CONFIG\_WLEN\_8 | UART\_CONFIG\_STOP\_ONE | UART\_CONFIG\_PAR\_NONE));

// enable UART interrupt

IntMasterEnable(); // enable processor interrupts

IntEnable(INT\_UART0); // enable the UART interrupt

UARTIntEnable(UART0\_BASE, UART\_INT\_RX | UART\_INT\_RT); //only enable RX and TX interrupts

/\* ADC SETUP \*/

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_ADC0);

ROM\_ADCHardwareOversampleConfigure(ADC0\_BASE, 32);

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\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_TIMER1);

ROM\_TimerConfigure(TIMER1\_BASE, TIMER\_CFG\_PERIODIC);

uint32\_t ui32Period = SysCtlClockGet() / 2;

// clock division by two to get 0.5 second delay

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

ROM\_IntEnable(INT\_TIMER1A);

ROM\_TimerIntEnable(TIMER1\_BASE, TIMER\_TIMA\_TIMEOUT);

ROM\_TimerEnable(TIMER1\_BASE, TIMER\_A);

while (1){

}

}

void Timer1AIntHandler(void)

{

uint32\_t ui32ADC0Value[4];

volatile uint32\_t ui32TempAvg;

volatile uint32\_t ui32TempValueC;

volatile uint32\_t ui32TempValueF;

uint32\_t i = 0;

uint8\_t temp\_str[10];

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

//moved everything inside while loop from main

ROM\_ADCIntClear(ADC0\_BASE, 1);

ROM\_ADCProcessorTrigger(ADC0\_BASE, 1);

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

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

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

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

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

// Output temperature via UART: "<temp> "

i = 0;

while(ui32TempValueF != 0){

temp\_str[i++] = (ui32TempValueF%10) + '0';

ui32TempValueF /= 10;

}

for(; i > 0; i--)

UARTCharPut(UART0\_BASE, temp\_str[i-1]);

UARTCharPut(UART0\_BASE, ' ');

SysCtlDelay(1000);

}

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

**Task 02:**

Youtube Link: https://www.youtube.com/watch?v=OAf1dUsxa58

**Modified Code:**

**#include** <stdint.h>

**#include** <stdbool.h>

**#include** "inc/hw\_memmap.h"

**#include** "inc/hw\_types.h"

**#include** "driverlib/gpio.h"

**#include** "driverlib/interrupt.h"

**#include** "driverlib/pin\_map.h"

**#include** "driverlib/sysctl.h"

**#include** "driverlib/adc.h"

**#include** "driverlib/rom.h"

// header files that enable UART interrupts

**#include** "inc/hw\_ints.h"

**#include** "driverlib/uart.h"

// header file to manipulate characters

**#include** <ctype.h>

**void** **UARTIntHandler**(**void**)

{

uint32\_t ui32Status;

uint8\_t letter; // received letter

uint32\_t ui32ADC0Value[4];

**volatile** uint32\_t ui32TempAvg;

**volatile** uint32\_t ui32TempValueC;

**volatile** uint32\_t ui32TempValueF;

uint32\_t i = 0;

uint8\_t temp\_str[10];

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

**UARTIntClear**(UART0\_BASE, ui32Status); // clear the asserted interrupts

**while**(**UARTCharsAvail**(UART0\_BASE)) // loop while there are chars

{

// case inversion for letters (lower->upper, upper-> lower)

letter = **UARTCharGetNonBlocking**(UART0\_BASE);

**if**(letter == 'G')

**GPIOPinWrite**(GPIO\_PORTF\_BASE, GPIO\_PIN\_3, GPIO\_PIN\_3);

**if**(letter == 'B')

**GPIOPinWrite**(GPIO\_PORTF\_BASE, GPIO\_PIN\_2, GPIO\_PIN\_2);

**if**(letter == 'R')

**GPIOPinWrite**(GPIO\_PORTF\_BASE, GPIO\_PIN\_1, GPIO\_PIN\_1);

**if**(letter == 'g')

**GPIOPinWrite**(GPIO\_PORTF\_BASE, GPIO\_PIN\_3, ~GPIO\_PIN\_3);

**if**(letter == 'b')

**GPIOPinWrite**(GPIO\_PORTF\_BASE, GPIO\_PIN\_2, ~GPIO\_PIN\_2);

**if**(letter == 'r')

**GPIOPinWrite**(GPIO\_PORTF\_BASE, GPIO\_PIN\_1, ~GPIO\_PIN\_1);

**if**(letter == 'T')

{

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

ROM\_ADCIntClear(ADC0\_BASE, 1);

ROM\_ADCProcessorTrigger(ADC0\_BASE, 1);

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

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

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

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

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

// Output temperature via UART: "<temp> "

i = 0;

**while**(ui32TempValueF != 0){

temp\_str[i++] = (ui32TempValueF%10) + '0';

ui32TempValueF /= 10;

}

**for**(; i > 0; i--)

**UARTCharPut**(UART0\_BASE, temp\_str[i-1]);

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

}

**UARTCharPutNonBlocking**(UART0\_BASE, letter); // echo character

}

}

T **int** **main**(**void**) {

/\* CPU Clock SETUP () \*/

**SysCtlClockSet**(SYSCTL\_SYSDIV\_4 | SYSCTL\_USE\_PLL | SYSCTL\_OSC\_MAIN | SYSCTL\_XTAL\_16MHZ);

/\* UART SETUP \*/

// enable UART0 & Port A peripherals

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_UART0);

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_GPIOA);

// configure Tx and Rx pins as UART pins

**GPIOPinConfigure**(GPIO\_PA0\_U0RX);

**GPIOPinConfigure**(GPIO\_PA1\_U0TX);

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

// LED SETUP as part of UART

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_GPIOF); // enable GPIO port for LED

**GPIOPinTypeGPIOOutput**(GPIO\_PORTF\_BASE, GPIO\_PIN\_1|GPIO\_PIN\_2|GPIO\_PIN\_3); // enable pin for LED PF2

// initialize UART parameters: 115200 Baud Rate, 8 data bits, 1 stop bit

**UARTConfigSetExpClk**(UART0\_BASE, **SysCtlClockGet**(), 115200,

(UART\_CONFIG\_WLEN\_8 | UART\_CONFIG\_STOP\_ONE | UART\_CONFIG\_PAR\_NONE));

// enable UART interrupt

**IntMasterEnable**(); // enable processor interrupts

**IntEnable**(INT\_UART0); // enable the UART interrupt

**UARTIntEnable**(UART0\_BASE, UART\_INT\_RX | UART\_INT\_RT); //only enable RX and TX interrupts

// output "Enter Text: "

**UARTCharPut**(UART0\_BASE, 'E');

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

**UARTCharPut**(UART0\_BASE, 't');

**UARTCharPut**(UART0\_BASE, 'e');

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

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

**UARTCharPut**(UART0\_BASE, 'C');

**UARTCharPut**(UART0\_BASE, 'm');

**UARTCharPut**(UART0\_BASE, 'd');

**UARTCharPut**(UART0\_BASE, ':');

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

/\* ADC SETUP \*/

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_ADC0);

ROM\_ADCHardwareOversampleConfigure(ADC0\_BASE, 32);

**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);

**while** (1) // let interrupt handler do the UART echo function

{

// if (UARTCharsAvail(UART0\_BASE)) UARTCharPut(UART0\_BASE, UARTCharGet(UART0\_BASE));

}

}

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