**Date Submitted: 10/11/18**

**Task 00: Execute provided code**

**#include** <stdint.h>

**#include** <stdbool.h>

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

**#include** "utils/uartstdio.h"

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

{

uint32\_t ui32Status;

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

{

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

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

**SysCtlDelay**(**SysCtlClockGet**() / (1000 \* 3)); //delay ~1 msec

**GPIOPinWrite**(GPIO\_PORTF\_BASE, GPIO\_PIN\_2, 0); //turn off LED

}

}

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

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

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

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

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

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

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

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

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

**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

**UARTStdioConfig**(0, 115200, **SysCtlClockGet**());

**UARTprintf**("Enter Text: \r\n");

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

{

}

}

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

**Task 01:**

Youtube Link: <https://www.youtube.com/watch?v=enlhQbYUDEo>

**#include** <stdint.h>

**#include** <stdbool.h>

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

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

**#include** "inc/tm4c123gh6pm.h"

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

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

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

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

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

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

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

**#define** TARGET\_IS\_BLIZZARD\_RB1

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

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

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

**#include** "utils/uartstdio.h"

// Holding the ADC values that come in FIFO

uint32\_t ui32ADC0Value[4];

// Variables for calculating temperature from the sensor

**volatile** uint32\_t ui32TempAvg;

**volatile** uint32\_t ui32TempValueC;

**volatile** uint32\_t ui32TempValueF;

**int** **main**(**void**)

{

// Set up the system clock at 40 MHz

ROM\_SysCtlClockSet(SYSCTL\_SYSDIV\_5|SYSCTL\_USE\_PLL|SYSCTL\_OSC\_MAIN|SYSCTL\_XTAL\_16MHZ);

// Enable ADC0 Peripheral

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_ADC0);

// Samples 32 times with 4 samples per time

ROM\_ADCHardwareOversampleConfigure(ADC0\_BASE, 16);

// Configure GPIO

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

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

// Configure ADC Sequencer 2

ROM\_ADCSequenceConfigure(ADC0\_BASE, 2, ADC\_TRIGGER\_PROCESSOR, 0);

// Configuring all 4 steps of the ADC sequencer

ROM\_ADCSequenceStepConfigure(ADC0\_BASE, 2, 0, ADC\_CTL\_TS);

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

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

// Sample Temperature sensor & configure interrupt flag

// Last conversion on sequencer 2

ROM\_ADCSequenceStepConfigure(ADC0\_BASE, 2, 3, ADC\_CTL\_TS|ADC\_CTL\_IE|ADC\_CTL\_END);

// Enable ADC sequencer 2

ROM\_ADCSequenceEnable(ADC0\_BASE, 2);

/\* TIMER1A Configurations \*/

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_TIMER1);

**TimerConfigure**(TIMER1\_BASE, TIMER\_CFG\_PERIODIC);

**TimerLoadSet**(TIMER1\_BASE, TIMER\_A, **SysCtlClockGet**()\*0.5);

**IntEnable**(INT\_TIMER1A);

**TimerIntEnable**(TIMER1\_BASE, TIMER\_TIMA\_TIMEOUT);

**IntMasterEnable**();

**TimerEnable**(TIMER1\_BASE, TIMER\_A);

/\* UART \*/

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

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

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

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

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

**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

**UARTStdioConfig**(0, 115200, **SysCtlClockGet**());

**while**(1)

{

}

}

**void** **Timer1IntHandler**(**void**)

{

// Clear the flag

ROM\_ADCIntClear(ADC0\_BASE, 2);

// Trigger the ADC Conversion

ROM\_ADCProcessorTrigger(ADC0\_BASE, 2);

// Wait for the conversion to Complete

**while**(!ROM\_ADCIntStatus(ADC0\_BASE, 2, false))

{

}

// Read the value from the ADC Sample Sequencer 2 FIFO

ROM\_ADCSequenceDataGet(ADC0\_BASE, 2, ui32ADC0Value);

// Calculate the Average temperature of sensor data

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

// TEMP in C = 147.5 - (( 75 \* (VREFP - VREFN) \* ADCVALUE) / 4096)

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

// Convert C to F => F = ( ( C \* 9 ) + 160 ) / 5

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

// print temperature

**UARTprintf**("This is current Temperature: %d degrees \r\n", ui32TempValueF);

**SysCtlDelay**(1000000);

}

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

**Task 02:**

Youtube Link: <https://www.youtube.com/watch?v=q40b0Di-hMs>

**#include** <stdint.h>

**#include** <stdbool.h>

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

**#define** TARGET\_IS\_BLIZZARD\_RB1

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

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

**#include** "utils/uartstdio.h"

// Holding the ADC values that come in FIFO

uint32\_t ui32ADC0Value[4];

// Variables for calculating temperature from the sensor

**volatile** uint32\_t ui32TempAvg;

**volatile** uint32\_t ui32TempValueC;

**volatile** uint32\_t ui32TempValueF;

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

{

uint32\_t ui32Status;

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

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

**char** characters = **UARTCharGet**(UART0\_BASE); // get the input

**UARTprintf**("%c", characters); // echo back the character

**if**(characters == 'R')

{

**GPIOPinWrite**(GPIO\_PORTF\_BASE, GPIO\_PIN\_1, GPIO\_PIN\_1); // Turn on Red LED

}

**else** **if**(characters == 'r')

{

**GPIOPinWrite**(GPIO\_PORTF\_BASE, GPIO\_PIN\_1, 0); // Turn off Red LED

}

**else** **if**(characters == 'G')

{

**GPIOPinWrite**(GPIO\_PORTF\_BASE, GPIO\_PIN\_3, GPIO\_PIN\_3); // Turn on Green LED

}

**else** **if**(characters == 'g')

{

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

}

**else** **if**(characters == 'B')

{

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

}

**else** **if**(characters == 'b')

{

**GPIOPinWrite**(GPIO\_PORTF\_BASE, GPIO\_PIN\_2, 0); // Turn off Blue LED

}

**else** **if**(characters == 'T') // Display the Temperature

{

// Clear the flag

ROM\_ADCIntClear(ADC0\_BASE, 2);

// Trigger the ADC Conversion

ROM\_ADCProcessorTrigger(ADC0\_BASE, 2);

// Wait for the conversion to Complete

**while**(!ROM\_ADCIntStatus(ADC0\_BASE, 2, false))

{

}

// Read the value from the ADC Sample Sequencer 2 FIFO

ROM\_ADCSequenceDataGet(ADC0\_BASE, 2, ui32ADC0Value);

// Calculate the Average temperature of sensor data

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

// TEMP in C = 147.5 - (( 75 \* (VREFP - VREFN) \* ADCVALUE) / 4096)

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

// Convert C to F => F = ( ( C \* 9 ) + 160 ) / 5

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

// print temperature

**UARTprintf**("\r\nThis is current Temperature: %d degrees \r\n", ui32TempValueF);

}

}

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

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

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

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

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

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

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

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

// Enable ADC0 Peripheral

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_ADC0);

// Samples 32 times with 4 samples per time

ROM\_ADCHardwareOversampleConfigure(ADC0\_BASE, 16);

// Configure GPIO

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

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

// Configure ADC Sequencer 2

ROM\_ADCSequenceConfigure(ADC0\_BASE, 2, ADC\_TRIGGER\_PROCESSOR, 0);

// Configuring all 4 steps of the ADC sequencer

ROM\_ADCSequenceStepConfigure(ADC0\_BASE, 2, 0, ADC\_CTL\_TS);

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

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

// Sample Temperature sensor & configure interrupt flag

// Last conversion on sequencer 2

ROM\_ADCSequenceStepConfigure(ADC0\_BASE, 2, 3, ADC\_CTL\_TS|ADC\_CTL\_IE|ADC\_CTL\_END);

// Enable ADC sequencer 2

ROM\_ADCSequenceEnable(ADC0\_BASE, 2);

**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

**UARTStdioConfig**(0, 115200, **SysCtlClockGet**());

**UARTprintf**("Enter The Command:\r\n");

**UARTprintf**("R: Turn on RED LED\r\n");

**UARTprintf**("r: Turn off RED LED\r\n");

**UARTprintf**("G: Turn on GREEN LED\r\n");

**UARTprintf**("g: Turn off GREEN LED\r\n");

**UARTprintf**("B: Turn on BLUE LED\r\n");

**UARTprintf**("b: Turn off BLUE LED\r\n");

**UARTprintf**("T: Display Temperature\r\n");

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

{

}

}

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